The Journal of Computing Sciences in Colleges

Volume 25, Number 5 **May 2010**

The Journal of Computing Sciences in Colleges

Papers of the Eighth Annual CCSC Mid-South Conference

March 26-27, 2010
Harding University
Searcy, Arkansas

Papers of the Sixteenth Annual CCSC Central Plains Conference

April 9-10, 2010
Park University
Parkville, Missouri

Volume 25, Number 5 **May 2010**

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 Jim Aman, CCSC Membership Chair, Saint Xavier University, Chicago, IL 60655.

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:

Myles McNally, President (2010), Webmaster, Professor of Computer Science, Alma College, 614 W. Superior St., Alma, MI 48801, (989) 463-7163 (O), (989) 463-7079 (fax), mcnally@alma.edu.

Bob Neufeld, Vice President (2010), Professor Emeritus of Computer Science, McPherson College, P. O. Box 421, North Newton, KS 67117, neufeld@mcpherson.edu.

Jim Aman, Membership Chair (2010), Assoc. Professor, Computer Science, Saint Xavier University, Chicago, IL 60655, (773) 298-3454 (O), (614) 492-1306 (H), (630) 728-2949 (cell), aman@sxu.edu.

Bill Myers, Treasurer (2011), Dept. of Computer Studies, Belmont Abbey College, Belmont, NC 28012-1802, (704) 461-6823, (704) 461-5051, (Fax), myers@crusader.bac.edu.

John Meinke, Publications Chair, (2012), Collegiate Associate Professor, UMUC Europe, US Post: CMR 420, Box 3668, APO AE 09063; (H) Werderstr 8, D-68723 Oftersheim, Germany, 011-49-6202-5 77 79 16 (H), 011 - 49-6221- 31 58 71 (fax), meinkej@acm.org.

Kim P. Kihlstrom, Southwestern Representative (2011), Associate Professor of Computer Science, Westmont College, 955 La Paz Road, Santa Barbara, CA 93108, kimkihls@westmont.edu.

Elizabeth S. Adams, Eastern Representative (2011), James Madison University - Mail Stop 4103, CISAT - Department of Computer Science, Harrisonburg VA 22807, 540-568-2745(fax), e-mail: adamses@jmu.edu.

Deborah Hwang, Midwestern Representative (2011), Dept of Electrical Engineering and Computer Science, University of Evansville, 1800 Lincoln Avenue, Evansville, IN 47722, (812) 488-2193 (O), (812) 488-2780 (fax), hwang@evansville.edu.

Scott Sigman, Central Plains Representative (2011), Associate Professor of Computer Science, Drury University, Springfield, MO 65802, (417) 873-6831, ssigman@drury.edu.

Ernest Carey, Rocky Mountain Representative (2010), Dean, College of Technology and Computing, #249, Utah Valley University, Orem, UT 84058-5999, (801) 863-8237 (O), (801) 318-6439 (cell), ECarey@uvu.edu.

Lawrence D'Antonio, Northeastern Representative (2010), Ramapo College of New Jersey, Computer Science Dept., Mahwah, NJ 07430, (201) 684-7714, ldant@ramapo.edu.

David R. Naugler, Midsouth Representative (2010), Computer Science, Southeast Missouri State University, One University Plaza, Cape Girardeau, MO 63701, (573) 651-2787, dnaugler@semo.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), Department of Computer Science, Sam Houston State University, Huntsville, Texas 77341-2090, (936)294-1571, mcguire@shsu.edu.

Brent Wilson, Northwestern Representative (2012), Database Administrator, George Fox University, 414 N. Meridian St., Newberg, OR 97132, (503) 554-2722 (O), (503) 554-3884 (fax), bwilson@georgefox.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, Collegiate Professor, UMUC Europe, US Post: CMR 420, Box 3669, APO AE 09063; (H) Werderstr 8, D-68723 Oftersheim, Germany. 011-49-6202-5 77 82 14, sdean@faculty.ed.umuc.edu.

Robert Bryant, Comptroller, Professor & Information Tech. Program Director, MSC 2615, Gonzaga University, Spokane, WA 99258, (509) 313-3906, bryant@gonzaga.edu.

Paul D. Wiedemeier, National Partners Program Coordinator, The University of Louisiana at Monroe, Computer Science and CIS Department, 700 University Avenue, Administration Building, Room 2-37, Monroe, LA 71209, 318-342-1856, (office), 318-342-1101 (fax), wiedemeier@ulm.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 a National Partner they are invited to participate in our regional conferences. Visit with their representatives there.

Microsoft Corporation

Cengage Learning

Shodor

Turings Craft

FOREWORD

And, once again two conferences have produced great programs which I'm proud to be a part of. As I look at the programs I really wish that I could attend both!

Getting this manuscript out is always a challenge. We have the conference committees trying to put programs together and then refining them, but we also have printing deadlines. My thanks to the Dan Brandon and Dean Sanders for their hard work feeding me manuscript that was carefully laid out by the respective conference committees. My thanks also to Dan and Dean for their responsiveness when I was "harassing" them that we had printing and shipping deadlines to worry about. The programs for both conferences are excellent, and I heartily commend the papers in this issue. Those able to attend the conferences will benefit greatly from the presentations and we will all benefit greatly from having the copy of the papers from both conferences available to us. This is particularly true for those of us unable to attend either conference.

Let me take a moment here to call your attention to our National Partners listed above. Their financial support enables us to continue to offer the CCSC regional conferences at an inexpensive registration rate, allowing those of us with minimal travel budgets to attend professional conferences. Their generosity is very much appreciated.

Let me also recognize Upsilon Pi Epsilon as a supporter of student programs at our regional conferences.

My thanks to all the supporters and facilitators, and it's a pleasure to present you with two more sets of excellent conference papers.

John Meinke
UMUC – Europe
CCSC Publications Chair

Papers of the Eighth Annual CCSC Mid-South Conference

March 26-27, 2010
Harding University
Searcy, Arkansas

WELCOME TO THE 2010 CCSC MID-SOUTH CONFERENCE

On behalf of the conference committee, I welcome you to the Eighth Annual Consortium for Computing Sciences (CCSC) Conference for the Mid-South region. We are pleased to be hosted this year by Harding University in Searcy, Arkansas.

The conference program includes eighteen refereed papers, five tutorials, two workshops, one panel discussion, and two sessions of oral presentations of undergraduate student research. We are also introducing our first Nifty Assignments session, and we continue to host our annual undergraduate student programming competition. Sessions include a broad spectrum of topics from the computing disciplines. We are especially pleased to offer multiple presentations in the areas of security, machine learning, robotics, computing curriculum, and networking.

The thirty-four papers submitted were reviewed using a double-blind process with at least three reviewers per paper. Eighteen papers were accepted for publication in the Journal of Computing Sciences in Colleges and for presentation at this year's conference. We would like to especially thank all of the authors for their hard work and high-quality submissions and all the reviewers for their careful reviews.

The Mid-South conference committee, the regional board, and the faculty and staff of the Harding University Department of Computer Science worked hard organizing this conference. I thank all of you for your varied contributions and the many hours of hard work that make this conference possible.

We hope you enjoy the conference and find it to be a valuable opportunity for professional development. We also look forward to seeing you at next year's CCSC Mid-South Conference hosted by the University of Central Arkansas in Conway.

Gabriel J. Ferrer, Hendrix College
Conference Chair

2010 MID-SOUTH CONFERENCE COMMITTEE

Gabriel Ferrer, Conference Chair . . . . . Hendrix College, Conway, AR
Tim Baird, Site Chair . . . . . Harding University, Searcy, AR
Larry Morell, Papers Co-Chair . . . . . Arkansas Tech University, Russellville, AR
Sean Geoghegan, Papers Co-Chair . . . . . University of Arkansas at Little Rock, AR
Jane Renwick, Papers Co-Chair . . . . . University of Arkansas at Fort Smith, AR
Otha Britton, Panels/Workshops/Tutorial Co-Chair University of Tennessee - Martin, TN
H. Conrad Cunningham, Panels/Workshops/Tutorial Co-Chair . . . . . University of Mississippi
Carl Burch, Nifty Assignments . . . . . Hendrix College, Conway, AR

Steve Baber, Student Programming Contest Co-Chair Harding University, Searcy, AR
David Hoelzeman, Student Programming Contest Co-Chair . . . . . Arkansas Tech University, Russellville, AR
David Middleton, Student Papers Co-Chair . . . . . Arkansas Tech University, Russellville, AR
Mark Goadrich, Student Papers Co-Chair . . . . . Centenary College of Louisiana, Shreveport, LA
Linda Sherrell, Publicity Chair . . . . . University of Memphis

REGIONAL BOARD — CCSC MID-SOUTH REGION

H. Conrad Cunningham, Regional Board Chair . . . . . University of Mississippi, University, MS
David Naugler, CCSC National Board Representative and Past Conference Chair . . . . . Southeast Missouri State University, Cape Girardeau, MO
Paul D. Wiedemeier, Membership and Registration Chair . . . . . University of Louisiana at Monroe, LA
Gabriel Ferrer, Conference Chair (2010) . . . . . Hendrix College, Conway, AR
Daniel Brandon, Editor . . . . . Christian Brothers University, Memphis, TN
David Hoelzeman, Webmaster . . . . . Arkansas Tech University, Russellville, AR
Linda Sherrell, Treasurer . . . . . University of Memphis, Memphis, TN
Tim Baird, Site Chair . . . . . Harding University, Searcy, AR
Otha Britton, Past Site Chair (2009) . . . . . University of Tennessee - Martin, TN
Larry Morell, Next Conference Chair (2011) . . . . . Arkansas Tech University, Russellville, AR
Vamsi Paruchuri, Next Site Chair (2011) . . . . . University of Central Arkansas, Conway, AR

REVIEWERS — 2010 CCSC MID-SOUTH CONFERENCE

Neal Gibson . . . . . Arkansas Department of Education, Little Rock, AR
David Hoelzeman . . . . . Arkansas Tech University, Russellville, AR
David Middleton . . . . . Arkansas Tech University, Russellville, AR
Larry Morell . . . . . Arkansas Tech University, Russellville, AR
Matt Brown . . . . . Arkansas Tech University, Russellville, AR
Roger Fang . . . . . Arkansas Tech University, Russellville, AR
Yingbing Yu . . . . . Austin Peay State University, Clarksville, TN
Jenq-Foung Yao . . . . . Chicago State University, Chicago, IL
Arthur Yanushka . . . . . Christian Brothers University, Memphis, TN
Dan Brandon . . . . . Christian Brothers University, Memphis, TN
Tebring Daly . . . . . Collin College, Plano, TX
John Stamey . . . . . Cosatal Carolina University, Conway, SC
Quinsulon Israel . . . . . Drexel University, Philadelphia, PA
Carl Burch . . . . . Hendrix College, Conway, AR
Gabriel Ferrer . . . . . Hendrix College, Conway, AR
John Ross . . . . . Indiana University Kokomo, Kokomo, IN
David Sonnier . . . . . Lyon College, Batesville, AR
Hira Herrington . . . . . Lyon College, Batesville, AR
Dennis Brylow . . . . . Marquette University, Milwaukee, WS
Jimmie Purser . . . . . Millsaps College, Jackson, MS
Cong-Cong Xing . . . . . Nicholls State University, Thibodaux, LA
Patrick McDowell . . . . . Southeastern Louisiana University, Hammond, LA
Theresa Beaubouef . . . . . Southeastern Louisiana University, Hammond, LA
David Naugler . . . . . Southeastern Missouri State University, Cape Girardeau, MO
Max North . . . . . Southern Polytechnic State University, Marietta, GA
Deborah Dunn . . . . . Stephen F. Austin State University, Nacogdoches, TX
Robert Strader . . . . . Stephen F. Austin State University, Nacogdoches, TX
Anne-Marie Eubanks . . . . . Stephen F. Austin State University, Nacogdoches, TX
Robert England . . . . . Transylvania University, Lexington, KY
Haifei Li . . . . . UniversityUnion University, Jackson, TN
Wesley Deneke . . . . . University of Arkansas, Fayetteville, AR
Wingning Li . . . . . University of Arkansas, Fayetteville, AR
John Talburt . . . . . University of Arkansas at Little Rock, Little Rock, AR
Kamilia Messaoudi . . . . . University of Arkansas at Little Rock, Little Rock, AR
Sean Geoghegan . . . . . University of Arkansas at Little Rock, Little Rock, AR
Serhan Dagtas . . . . . University of Arkansas at Little Rock, Little Rock, AR
Soukaina Messaoudi . . . . . University of Arkansas at Little Rock, Little Rock, AR
Rick Massengale . . . . . University of Arkansas at Fort Smith, Fort Smith, AR
Janet Renwick . . . . . University of Arkansas at Fort Smith, Fort Smith, AR
Greg Holland . . . . . University of Arkansas at Little Rock, Little Rock, AR
Jose Cordova . . . . . University of Louisiana at Monroe, Monroe, LA

Paul Wiedemeier . . . . . University of Louisiana at Monroe, Monroe, LA
Chris Simmons . . . . . University of Memphis, Memphis, TN
Jake Qualls . . . . . University of Memphis, Memphis, TN
Linda Sherrell . . . . . University of Memphis, Memphis, TN
Conrad Cunningham . . . . . University of Mississippi, Oxford, MS
Lakisha Simmons . . . . . University of Mississippi, Oxford, MS
Jaime Nino . . . . . University of New Orleans, New Orleans, LA
Art Shindhelm . . . . . Western Kentucky University, Bowling Green, KY
Andrea Edwards . . . . . Xavier University of Louisiana, New Orleans, LA
Kun Zhang . . . . . Xavier University of Louisiana, New Orleans, LA
Hyoil Han . . . . . LeMoyne-Owen College, Memphis, TN
Cen Li . . . . . Middle Tenn State University, Murfreesboro, TN
Roland Untch . . . . . Middle Tenn State University, Murfreesboro, TN
Anatole Ruslanov . . . . . State University of New York at Fredonia, Fredonia, NY
Milam Aiken . . . . . University of Mississippi, Oxford, MS

CONCURRENT PROGRAMMING IN ERLANG*

PRE-CONFERENCE WORKSHOP

David R. Naugler
Southeast Missouri State University
One University Plaza
Cape Girardeau
573-651-2787
dnaugler@semo.edu

Erlang is not just another general purpose functional programming language and it is not the best or most appropriate functional language for all uses. There are many worthy programming languages and the best or most appropriate depends on the intended uses. Erlang is currently the best and most appropriate functional language for writing robust, scalable, programs that can effectively use a one core processor, multicore processors , multiple processors, or trusted networks Concurrent programs in Erlang can keep running when more processors or core are added, when some fail or are removed and can be written to allow hot code swapping so updating does not even require stopping the process being updated. This can be done using significantly less code and done faster than in other languages. Erlang was designed for concurrency. The programmer can concentrate on what the processes are supposed to do and let them fail if a problem occurs. Failure of a process is a normal response, not a problem to be avoided. Much less defensive coding is required and fault tolerance is much easier to achieve. This is not magic, it is using a tool designed for the job. The Erlang programming model is to use many very lightweight processes which do not use shared memory, locks, or synchronized methods and which communicate by message passing. The processes are controlled the Erlang runtime and not by the operating system. The programmer's focus is on what the program is supposed to do and not on what can go wrong.

Erlang is tested, and documentation, source code, and Windows and Linux implementations are freely available. There are two excellent books [1], [2]. It has been generally available for over a decade years and has been used for large industrial applications. It interfaces well with other languages. You many well have already used software which has components written in Erlang. . Implementations for Linux and Windows are readily available [1], there is good documentation, an Erlang

Erlang programming requires a different, but natural, way of thinking about and implementing concurrent systems. It is relatively easy to learn and can be fun. It is not the total solution to concurrent programming, it many not be the language that dominates

---

but Erlang will be almost certainly greatly influence the language or languages that do dominate.

This workshop is a hands-on introduction to concurrent programming in Erlang. An introductory knowledge of Erlang or experience with a modern functional language is assumed.

[1] Armstrong, J., *Programming Erlang:Software for a Concurrent World*, Raleigh, NC: The Pragmatic Bookshelf, 2007.

[2] Cesarini, F., Thompson, S., *Erlang Programming*, Sebastobol,CA: O'Reilly, 2009

David Naugler is a professor at Southeast Missouri State University where he has taught since 1981. He has long had a special interest in functional programming languages, most currently in Erlang. He also has an interest in Computational Science. He has given workshops in Erlang at the introductory and intermediate level in which many aspects of sequential Erlang programming have been covered at previous CCSC-MS conferences.

AN INTRODUCTION TO OPENGL GRAPHICS PROGRAMMING*

PRE-CONFERENCE WORKSHOP

Otha L. Britton
University of Tennessee at Martin
Martin, Tennessee 38238
731-881-7587
britton@utm.edu

In order to display computer graphics from a language such as C or C++, a graphics application programmers' interface must be used. The most popular multi-platform API for this is the OpenGL graphics system. Even the complex movie generation systems such as the academy award winning Maya software, used for such computer animation movies as Toy Story, is based on OpenGL. OpenGL, along with an appropriate windowing toolkit such as GL Utility Toolkit (GLUT), can be used to effectively display computer graphics. Students can be taught easily to display windows and make elementary drawings using only the basics of OpenGL.

The workshop will start with a quick overview of how to set up various integrated development environments (IDEs), such as MS Visual C++ Express 2008 and Bloodshed's Dev-C++, to use OpenGL and GLUT. This will be followed by a basic introduction to programming with the OpenGL graphics API. A brief discussion of the OpenGL coordinate systems and simple 3D drawing techniques will be given. The participants will then be shown how to use the scaling, rotation, and translation functions as well as functions for processing mouse and keyboard events. Then the fun will begin with an introduction to computer animation. Examples that will be constructed begin with the standard rotating cube, then the bouncing ball, and will conclude with a 3D walking robot, which utilizes the hierarchical structure to achieve appropriate motion.

The language used will be C++, but OpenGL can be used in many different languages. All of the software utilized (except for the operating system) is free for students and can be incorporated easily into beginning as well as advanced computer programming courses. The workshop presenter has taught a course in computer graphics for many years and has introduced the basic graphics concepts into a second-semester C++ course. The introduction of graphics into such a course has spurred much interest in most of the students and even the poorer students have seemed to perk up and do high-quality work on the related assignments. Some textbooks are now available that

utilize an IDE to permit basic Windows programming early in a programming course, but using OpenGL instead may make drawing and event-driven programming easier to understand.

This workshop is a hands-on introduction to programming with OpenGL. A basic knowledge of the C++ language is assumed, although those with Java programming knowledge should be able to quickly adjust and follow along.

FOCUSED MULTI-DOCUMENT SUMMARIZATION: HUMAN SUMMARIZATION ACTIVITY VS. AUTOMATED SYSTEMS TECHNIQUES*

Quinsulon L. Israel
Drexel University
Philadelphia, PA 19104
(215) 397-4317
qisrael1906@acm.org

Hyoil Han
LeMoyne-Owen College
Memphis, TN 38126
(901) 435-1391
hyoil.han@acm.org

Il-Yeol Song
Drexel University
Philadelphia, PA 19103
(215) 895-2489
song@drexel.edu

ABSTRACT

Focused Multi-Document Summarization (MDS) is concerned with summarizing documents in a collection with a concentration toward a particular external request (i.e. query, question, topic, etc.), or focus. Although the current state-of-the-art provides somewhat decent performance for DUC/TAC-like evaluations (i.e. government and news concerns), other considerations need to be explored. This paper not only briefly explores the state-of-the-art in automatic systems techniques, but also a comparison with human summarization activity.

1. INTRODUCTION

Multi-document summarization aims to create a compressed summary of a collection of documents, while retaining the main characteristics and pertinent

information within those documents. Adding a focus on this process creates a summary that also meets a specific request; hence, focused MDS. Extractive summarization uses whole sentences from the documents in the collection that "inform" the most regarding a specific request. However, abstractive summarization entails fusing bits and pieces of information from various sentences in the document collection into semantically similar information represented in a new way.

The process of summarization can be described in the following steps: 1) Process sentences using sentence boundary detection, optionally using stop word removal, tokenization, and other light syntactic parsing, 2) Score each sentence according to heuristically or automatically ascertained features: terms of the sentence, position of the terms and/or sentence, relevance with focus query, derived concepts, etc., 3) Rank and select the most salient and informative of those scored sentences according to some linear combination of features

Why do we need such an automated process? The burden of extensive amounts of information (i.e. information overload), so readily available in digital form, has placed a heavy cognitive burden (information overload) on society. It has become even more important for a user to be able to quickly read, understand and utilize digital information as easily as they can access it. For instance, fast generation of result snippets with search terms within context has been done in Google for search results [22]; however, a condensing of the actual text linked to each search result may distinguish better usefulness.

Focused MDS has a wide range of usefulness and is applicable across many domains. Considering the myriad of different types of texts publicly available, especially within an immensely large and ever growing corpus such as the World Wide Web, summarization representations and techniques need to be as robust and as efficient as possible, in order to be useful in the average laymen's everyday life.

The remainder of this paper is organized as follows: Section 2 discusses the process of human summarization activity with some historical background, Section 3 discusses sentences processing in automatic systems and its use of linguistic techniques, Section 4 discusses the representations of documents and they facilitate MDS processing, Section 5 discusses how position information improvements sentence scoring, Section 6 discusses sentence scoring and ranking and using N-gram statistics versus language modeling and classification versus clustering, Section 7 discusses sentence selection and how the issues of redundancy, compression and coherence effect choosing the most salient sentences for a summary, and finally, Section 8 concludes this work.

2. THE INFLUENCE OF HUMAN SUMMARIZATION ACTIVITY

Human agreement studies have been performed since the 1960's [8], where there was little agreement between sentences extracted by machines using frequency expectation and those extracted by humans. For single document summarization, studies reported that 79% of the sentences in a human-generated abstract were a "direct match" to a sentence from a source document [10]. Therefore, it is natural that many current MDS (Multi-Document Summarization) systems have been made to be extractive, attempting to mimic this human behavior. However, for MDS, it has been shown that no more than 55% of the vocabulary contained in human-generated abstracts can be found within the source documents [5], signifying a move to a more abstractive means of summarizing for this task. Also, different people choose different content for their summaries [9, 12, 16]. This is also true for the human judges (usually four) creating model summaries in DUCs/TACs, and a correlation of automatic scores and these human model summaries is taken into account each year [6] to see how well automatic systems are improving, or not. Even multiple human summaries on the same collection of documents often do not have much agreement, with unigram overlap of only 40% in a study from Lin and Hovy [11]. In fact, in order for a consensus summary to be created from single document summaries from different humans, there must be as many as 30-40 summaries gathered before a consensus becomes stable [21].

3. SENTENCE PROCESSING

Processing a document can often involve several layers depending on the system's task and strategy, as well as the representation, structure and syntactic variations of the document. Once processing is complete, the document is represented in a form in which the system can interface. Normally, "syntactic parsing" is performed at the most atomic level: each token of these sentences are "tagged" by machine-readable codes according to the grammatical rules and context of its respective language for part-of-speech (POS) [4].

The subjects, predicates and objects can also be parsed as terms or phrases into their semantic roles, such as pred(sub, obj). The functional relationships have started to gain more attention for summarization. This involves semantic role labeling and can be done with the aid of the well known and expensive PropBank [14], a collection of annotated propositions in predicates-argument form, or the freely available FrameNet [2], a collection of semantic elements of logical units centered on actions in frame form. By examining the roles of various elements in a sentence, a system can make better decisions on what exactly to compare. In fact, Nenkova et al. [12] used semantic annotation in their methodology to count occurrences of semantically equivalent content between summaries and the corpus, but only used shallow parsing to determine the respective important words. Consequently, they discovered factors that influence human judgments and used

them to create their system, only to increase the accuracy of content counting during evaluation and distribution building. Wang et al. [24] used pair-wise sentence semantic similarity. For each sentence, a "frame" was created. Next, WordNet was used to discover the semantic relations of all terms in the first frame with those in the second to determine if there is a semantic relation (e.g. synonym, hypernym, etc.). However, this was solely used for their similarity metric. On the other hand, Ouyang et al. [13] used NER (Named Entity Recognition) and NER counts, informative words, semantic similarity using WordNet Lesk function, etc., but do not use deeper semantic parsing for comparisons between semantic roles of different sentences.

The order of linguistic processing and the depth at which it is performed is a touchy subject. For summarization tasks, it appears it is best to use "shallow" analysis techniques that do not require heavy computing resources and trained deep parsers. Mostly all research described in the previous sub-sections use shallow techniques such as POS tagging and NER. This has been the normal standard of operation; however, systems such as in Shi et al. [19] have started to use "deeper" syntactic analysis as a basis for nominal semantic role labeling.

4. DOCUMENT (TEXT) REPRESENTATIONS

What is meant by the representation of a document? Document representation is defined here as a tangible, either visual or physical, in our case digital, formalism of the concrete (text) and abstract (meaning) properties embodying the document. It is a finite way to automatically organize and store in some numeric form (index), and then later, retrieve the document for a system to interface with it (e.g. as in early information retrieval). Document representations must be applied to smaller bits of information such as user queries, questions, and more importantly, sentences in order to be useful toward MDS.

One of the fundamental and most efficient ways of representing a document was the Vector Space Model (VSM) [17]. In the VSM, a subset of terms appearing within the document at least once are placed in a weighted, linear combination of vectors of terms to represent the document; when all unique terms present in the document are used, it is also known as the "bag-of-words" model [18]. The term frequency and inverse document frequency of the term are multiplied together to determine the final weight for that term, and then, usually normalized. This allows for geometric calculations to be performed with the vectors of other documents or sentences for similarity. An information need such as a user query vector or a given topic vector can be compared against the document vector as well, for specific focus. The VSM became the de facto standard in text representation because of its simplicity and performance in large text collections, and is the basis of early summarization methods.

5. DOCUMENT POSITION

Until recently, the information provided by the structure of documents, or the hierarchy of its units of varying granularity, has mostly not been leveraged effectively. Research such as [13] for DUC 2005-2006 data has shown that sentence position can be of benefit when choosing the most informative sentences. The use of sentence position in an extractive method of summarization is intuitive since important information usually is placed in specific locations within a well-formed, but unstructured text based, formal document such as those found in DUC/TAC conferences. Since these corpora are normally composed of news articles, the importance of information is consistently related to how close it is to the beginning of the document, as reported by and found in the work of Yih et al. [26] involving word positions; news reporting seeks to provide the most salient information as quickly as possible. In fact, it is important to note that baseline systems in DUC/TAC and other government sponsored evaluations used the first N words in documents, the first (and sometimes last) sentences of every document, the first N sentences of the most recent document, or some other derivations of lead words/sentences to create baseline summaries. These automatic systems often outperformed peers. Hence, researchers have begun to use the same position information, concentrating on enhancing ROUGE scores on past DUC evaluation data.

There have also been studies of using surface level linguistic cues such as transition (connecting or "cue") terms [20] that identify changes in topic on both a document and a segment level. These transition terms can be weighted in a meaningful way to help improve focused judgments towards informative sentences. For instance, in work from Sun et al. [20], it was shown that "cue" words could identify topic shift in a single document as well as help identify the reversal of sub-topic order in a document compared to another, when the focus was on globally shared topics among documents.

6. SENTENCE SCORING AND RANKING

6.1 Simple Word (N-gram) Statistics vs. Statistical Language Modeling

Scoring for summarization involves calculating a numeric value for the significance of a sentence. However, of the utmost importance for focused summarization is scoring with an external factor besides sentence-to-sentence comparisons; not only do sentences need to be compared for similarity to each other, they also need to be skewed toward the user focus. After the score is calculated, the sentences are ordered from the highest score to the lowest. The most basic summarization, besides heuristic sentence choice based on position, is to count word-for-word matches of the terms of the focus input (query, question, or topic description) to the terms in sentences from the input collection. The

simplest approach for frequency-based probabilities is in [12] where summation of within-document content word probabilities was empirically most performant.

Ouyang et al. [13] tested both frequency of occurrence and binary appearance on the DUC 2005 data in an indirect manner, using human reference summaries to learn linear feature combinations through Support Vector Regression, and on the DUC 2006 data for direct testing. Results show that frequency of occurrence was more performant for actual scoring estimation than binary appearance measures. However, the work of [4] showed that using simple summation of the binary appearances of topic terms along with so-called signature terms, to create an "approximate oracle score," can improve sentence scoring; signature terms for a topic under observation are believed to be those important terms derived from a sub-corpus of related documents but are used less in the rest of the corpus. More sophisticated than simply counting terms (or phrases) is the use of the cosine similarity measure on the vectors of these terms. Cosine similarity is a fast and efficient means to score sentences in terms of relatedness to each other; however, to add focus to such a process, it is necessary to also compare the focus input vector with those sentences. This is done in [25] by adding query-sensitive similarity to the centroid of two documents with a metric similar to cosine similarity but with the addition of a complex function on the query terms. In [3], the cardinality of overlap in topic terms is used in their linear feature combination function, along with cosine similarity between only the title of the topic in the DUC 2006 task and an individual sentence.

In fact, the work of Arora et al. [1] used the latent dirichlet allocation (LDA) model with estimated probabilities based on the corpus to create multiple summaries, and then, automatically chose the best. Their framework assumes that a complete sentence of a document belongs to only one topic. Performance was reportedly better than the two top systems in DUC 2002 for the ROUGE-1 (unigram) value.

6.2 Classification vs. Clustering

As stated by Ouyang et al. [13], classification based models are usually adopted to solve discrete problems; therefore, they are imprecise against continuous real-value functions like linear combination sentence scoring (using features). In any classification method, an item either belongs to a particular class, or it does not. However, more categorical delineations can be determined for the sentences if a real value can be calculated for each observation. As such, clustering has proven more performant; similar documents are gathered together according to the weight of their cosine similarity metrics, and the document that has the highest similarity in its cluster should contain the most worthy sentence for extraction into the summary about that topic. This process has been used on a sentential level as well, as in [23]. Wan and Yang [23] cluster the sentences of a document using various methods: k-means, agglomerative, and divisive. Using these clustering methods they were able to find the clusters within the document

that truly represented sub-topics, and then extract the best similar sentence from each cluster centroid. This approach was similar to the MEAD system [15], which popularized the use of the document centroid for choosing the most salient sentence from a document. In this way, Wan and Yang [23] were able to choose the most salient yet diverse sentences. Also, terms can be clustered in order to help differentiate topics in a document as well [20]. The terms in each cluster can be associated by semantic similarity or by the entropy between different documents or different segments of documents.

Matrix representations of scoring equations add an efficient means of performing calculations due to the ability to use special manipulations such as in place normalization and finding eigenvectors. In fact, [24] use symmetric non-negative matrix factorization to calculate similarity to group similar sentences into clusters. In [3], a matrix was used to compute the hamming weights of terms in different sentences; the concern was to increase the significance of a sentence by the amount of pairs of significance words found. This hamming weight is then multiplied by the significant word count in segments and by word frequencies.

Using the techniques described above gives good results in terms of either speed of computation (cosine similarity in a low-dimensional degree) or better accuracy (matrix calculations). Clustering methods have shown a two-fold purpose in that they not only group sentences according to similarity with each other and a topic, but also delineate multiple sub-topics. It is also important to note that matrix calculations, though complex, also cluster sentences but have not been used specifically to delineate topics/subtopics to the authors' knowledge. Also, more studies regarding the use of combining very simplistic n-gram (bigram, named entities, phrases) frequencies and probabilities with text unit clustering (or matrix use) are needed to be carried out in depth.

7. SENTENCE SELECTION: REDUNDANCY, COMPRESSION, AND COHERENCE

Sentence selection is the second most important step for processing sentences as this step actually adds sentences to the summary. Its consideration in MDS should not be based solely on having the highest score, but also on sentence length, redundancy removal, compression and coherence. Many sentences may be important; however, the best among those containing redundant information must be selected.

Clustering via a cosine similarity metric is normally the method that has been used to group similar sentences in order to choose the sentence closest to the centroid and make subsequent choices from other clusters, as in [13] where maximum marginal relevance (MMR) was used with a threshold of 0.60. Also, Wan and Yang [23] used a variant of MMR between sentences to eliminate redundancy, whereas [24] used their own semantic similarity score. However, [4] used a pivoted QR algorithm instead of MMR

and reported better performance because of it. Pseudo relevance feedback, which uses sparse information to retrieve more information from a corpus, was also used by [4] to help reduce redundancy.

Sentence Compression may allow more sentences to be chosen into a summary. Yih et al. [26] eliminated syntactic units based on predefined heuristic templates and added these new, modified sentences to the pool along with their original counterparts. During sentence selection, the best sentence between the original sentence and its modifications are automatically chosen; this helps alleviate the problems of over-simplification. It is also important to note that simplification can either help or harm the coherence of system summaries depending on the techniques used: adverb removal, parenthetical phrase removal, phrases within commas, etc.

As mentioned in [7], coherence is a key factor that affects a judge's perception of readability for automatic summaries. [19] used longest common subsequences (LCS) of the sentence to be chosen with the sentence previously chosen for the summary, to try to maintain coherence. However, the weighting toward LCS is a tradeoff with attaining the highest similarity to a particular search focus.

Although there are many considerations to be made for sentence selection, it appears from the literature that there have not been many studies concerned with coherence. Coherence has begun to improve slightly during the DUC/TAC evaluations, but it appears to be mostly a byproduct of improved sentence ranking. Different concerns that can affect coherence such as topic segmentation, document structure, developing a scheme for information ordering, using cue words, etc. need to be studied.

8. CONCLUSION

Along with an explanation of multi-document summarization and its benefits, we have explored briefly some of the most novel techniques for improvement of MDS from the most promising and recently published research. Although the state-of-the-art of focused multi-document summarization has seen quite some improvement within the last decade, it has been shown that there is still room for further experimentation, using knowledge gained from the previously outlined human summarization activity; sentence processing, scoring, ranking and selection; and document representations and term position information. Human summarization activity was explored from the angle of instability in sentence consensus to the stability of salient words consensus. The subject of sentence processing and related activities was explored from light syntactic processing and semantics to term scoring and modeling techniques to the use of complex matrix calculations and the use of position information. A few of the previously lesser used, but important techniques were discussed; sentence compression and coherence improvement techniques are described in the context of the most simplistic yet powerful use.

REFERENCES

[1] Arora, R. and B. Ravindran, Latent dirichlet allocation based multi-document summarization, *Proceedings of the second workshop on Analytics for noisy unstructured text data*, 303, 91-97, 2008.

[2] Baker, C.F., C.J. Fillmore, and J.B. Lowe, The Berkeley FrameNet Project, *Proceedings of the 17th international conference on Computational linguistics,* 1, 86-90, 1998.

[3] Boudin, F. and J.M.T. Moreno, *NEO-CORTEX: A Performant User-Oriented Multi-Document Summarization System*, in *Proceedings of the 8th International Conference on Computational Linguistics and Intelligent Text Processing*, 551-562, 2007.

[4] Conroy, J.M., J.D. Schlesinger, and D.P. O'Leary, Topic-focused multi-document summarization using an approximate oracle score, *Proceedings of the COLING/ACL on Main conference poster sessions*, 152-159, 2006.

[5] Copeck, T. and S. Szpakowicz, Vocabulary Agreement Among Model Summaries and Source Documents, *ACL Text Summarization Workshop*, 2004.

[6] Dang, H.T. Overview of the DUC 2005, *Document Understanding Conference*, 2005.

[7] Dang, H.T. Overview of the DUC 2006, *Document Understanding Conference*, 2006.

[8] G. J. Rath, A. Resnick, and T.R. Savage, The formation of abstracts by the selection of sentences. Part I. Sentence selection by men and machines. *American Documentation,* 12(2), 139-141, 1961.

[9] Halteren, H.v., New Feature Sets for Summarization by Sentence Extraction, *IEEE Intelligent Systems,* 18(4), 34-42, 2003.

[10] Kupiec, J., J. Pedersen, and F. Chen, A trainable document summarizer, *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, 68-73,1995.

[11] Lin, C.-Y. and E. Hovy. Manual and Automatic Evaluation of Summaries, *Document Understanding Conference*, 2002.

[12] Nenkova, A., L. Vanderwende, and K. McKeown, A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization, *Proceedings of the 29th Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, 573-580, 2006.

[13] Ouyang, Y., S. Li, and W. Li, Developing learning strategies for topic-based summarization, *Proceedings of the sixteenth ACM Conference on information and knowledge management*, 79-86, 2007.

[14] Palmer, M., D. Gildea, and P. Kingsbury, The Proposition Bank: An Annotated Corpus of Semantic Roles, *Computational Linguistics*, 31(1), 71-106, 2005.

[15] Radev, D.R., H. Jing, and M. Budzikowska, Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies, *NAACL-ANLP 2000 Workshop on Automatic summarization,* 4, 21-30, 2000.

[16] Radev, D.R., Teufel, S., Saggion, H., Lam, W., Blitzer, J., Qi, H., Çelebi, A., Liu, D., Drabek, E., Evaluation challenges in large-scale document summarization, *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics,* 1, 375-382, 2003.

[17] Salton, G., A. Wong, and C.S. Yang, A Vector Space Model for Automatic Indexing, *Communications of the ACM,* 18(11), 613-620, 1974.

[18] Salton, G., Singhal, A., Mitra, M., Buckley, C., Automatic text structuring and summarization, *Information Processing & Management*, 33(2), 193-207, 1997.

[19] Shi, Z., Shi, Z., Melli, G., Wang, Y., Liu, Y., Gu, B., Kashani, M., Sarkar, A., Popowich, F., Question Answering Summarization of Multiple Biomedical Documents, *Proceedings of the 20th conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, 284-295, 2007.

[20] Sun, B., Prasenjit, M., Hongyuan, Z., Lee, G., John, Y., Topic Segmentation with Shared Topic Detection and Alignment of Multiple Documents, *Special Interest Group on Information Retrieval*, 2007.

[21] Teufel, S. and H.v. Halteren, Evaluating Information Content by Factoid Analysis: Human Annotation and Stability, *Empirical Methods in Natural Language Processing*, 2004.

[22] Turpin, A., Turpin, A., Tsegay, Y., Hawking, D., Williams, H. E., Fast generation of result snippets in web search, *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 127-134, 2007.

[23] Wan, X. and J. Yang, Multi-document summarization using cluster-based link analysis, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 299-306, 2008.

[24] Wang, D., Wang, D., Li, T., Zhu, S., Ding, C., Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 307-314, 2008.

[25] Wei, F., Li, W., Lu, Q., & He, Y., Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization, *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 283-290, 2008.

[26] Yih, W.-T., Goodman, J., Vanderwende, L., Suzuki, H., Multi-document summarization by maximizing informative content-words, *International Joint Conferences on Artificial Intelligence*, 1776-1782, 2007.

USING INFORMATION FLOW TO ANALYZE GRAMMARS *

Dr. Larry Morell
Computer and Information Science
Arkansas Tech University
Russellville, AR 72801
(479) 968-0355
lmorell@atu.edu

Dr. David Middleton
Computer and Information Science
Arkansas Tech University
Russellville, AR 72801
(479) 968-0628
dmiddleton@atu.edu

ABSTRACT

This paper presents algorithms for analyzing grammars using information flow in a graph. A grammar problem is modeled by an annotated graph. Information is then propagated around the graph and the problem solution is extracted from the graph's final state. By using this approach students need learn only one generic algorithm (information flow analysis), which they then can apply to several problems Teachers are able to develop problems to be solved directly from the graphs, thereby simplifying the task of generating grammars with required properties for homework and exams.

INTRODUCTION

When building a parser one must determine what terminals can begin or can follow a non-terminal.[1] The algorithms for solving these problems are well-known, but are somewhat difficult for many undergraduate students to comprehend, learn and apply. They are particularly difficult to remember because each seems to have its own trick and there appears to be no uniformity among the algorithms.

We present a uniform approach for solving all the problems based on the concept information flow. Grammar problems are modeled by annotated graphs. Information relevant to solving the problem is then propagated around the graph, using a standard algorithm. Once propagation completes, the solution to the original problem is extracted

from the final state of the graph. Using one generic algorithm simplifies the task of learning to solve the collection of problems is simplified.

INFORMATION FLOW ANALYSIS

When a solving a problem involves propagating information through a network of nodes, information flow analysis may be appropriate. In *information flow analysis* nodes are annotated with initial information. This information is then propagated to neighboring nodes, updating their information, which may cascade to further nodes.

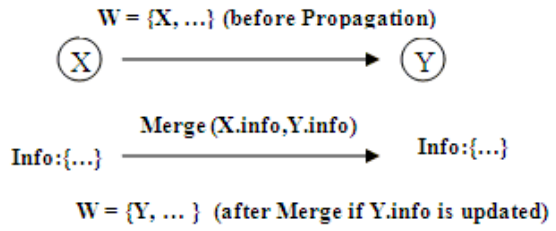
There are many problems that can be modeled by annotated graphs and solved by information flow. Classic problems such as determining the shortest path between two nodes fit the paradigm for information flow analysis. A variant of information flow analysis called *data flow analysis* [2] to check programs for various anomalies such as consecutive assignments to a variable without an intervening use, use of an undefined variable, and unused variables. Inherited and synthesized attributes in trees (attribute grammars) can be characterized by information flow analysis and form the basis for syntax-directed translation schemas [3].

Information flow algorithms consist of three basic steps:

1. Model: Build a graph that represents the problem to be solved. Problem information may be represented by annotated nodes or annotations
2. Propagate: Merge annotations at each node with the annotations of connected nodes. In many algorithms propagating may be as simple as forming the union of the two sets. Continue this merging process until the annotations won't change.
3. Extract: Extract the solution to the original problem from the graph

Propagation is illustrated by diagram 1. Propagation begins by extracting a node X from a working set W. Merge updates the information associated with each successor node Y using the information found from X. Y is added to the working set W if it is updated. The algorithm terminates when W is empty, indicating propagation is impossible.

Diagram 1



Here is a generic algorithm for solving problems in this way using forward propagation:

1. Build a graph to represent the problem with each node X annotated with X.info.
2. Initialize W to contain a set of nodes that have special characteristics (sources).
3. While W is not empty
   a.. Remove a single node X from W
   . b. For each successor Y of X
      i. Merge the information found in X.info with Y.info
      ii. Add Y to W when changes to Y.info require further propagation.
4. Extract the solution to the problem from the graph.

Implementing W as a stack yields depth-first propagation; implementing it as a queue yields to breadth-first propagation. The details of the algorithm vary across problems, but the general structure remains fixed.

GRAMMAR ANALYSIS VIA INFORMATION FLOW ANALYSIS

In this section we discuss some traditional grammars analysis algorithms using information flow analysis. Three examples are central to parsing: which non-terminals can derive a null string, which terminals can start string derived from a non-terminal, and which terminals can follow a non-terminal in a derivation.

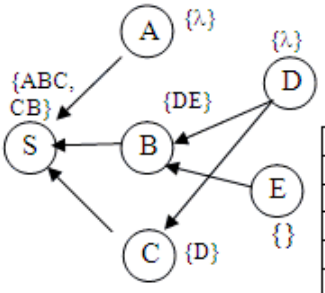
Each algorithm is defined in three stages: (1) map the grammar to an annotated graph, (2) propagate the annotated information around the graphs, and (3) extract the solution from final state of the graph.

We use uppercase Latin letters to represent arbitrary non-terminals from a set N and lowercase Latin letters to represent arbitrary terminals from a set T. Lowercase Greek letters are used to represent a (possibly empty) string of terminals and non-terminals. If we have a grammar rule $A \rightarrow \beta$, then we say β is a right-hand side associated with A. If we begin with an arbitrary string of terminals and non-terminals, α, and derive another string of terminals and non-terminals, β ,by repeatedly replacing a non-terminal in α with a right-hand side associated with that non-terminal in zero or more steps, we write $\alpha \Rightarrow^* \beta$. We use λ to represent the empty string.

Determining the non-terminals that derive λ is our first algorithm. We wish to compute: DL[X] = true if and only if $X \Rightarrow^* \lambda$. We begin by setting up the graph:

1. For each A in N, create a corresponding node labeled A in the graph.
2. For each rule $A \rightarrow X_1 X_2 \ldots X_n$ (each X_i in N)
   a. For each X_i, add arc $X_i \rightarrow A$ to the graph
   b. Add the string $X_1 X_2 \ldots X_n$ to the A.info (unless some X_i = A).When $A \rightarrow \lambda$, this places λ into A.info

Diagram 2




```
S -> ABC | CB
A -> a | lambda
B -> DE | b
C -> D | c
D -> d | lambda
E -> EB | e
```

X	DL[X]
S	False
A	True
B	False
C	True
D	True
E	False

Diagram 2 illustrates the graph that would be constructed for a particular grammar.

A is linked to B in the graph if A appears in the right-hand side of some grammar rule for B. Since it is impossible for grammar rule to contain a terminal and be used in a derivation that yields the empty string, we only need consider grammar rules whose right-hand side consists only of zero or more non-terminals. Similarly, self-loops indicating recursion are excluded since a recursive rule can only be used to derive lambda if the non-terminal derives lambda by some other rule. A.info is the set of all its associated right-hand sides under these restrictions.

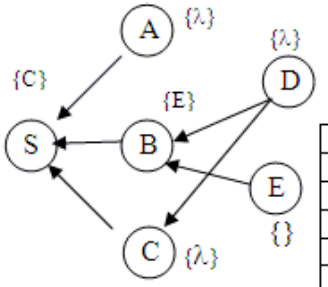
1. Initialize W it to contain all non-terminals A for which A directly derives λ.

2. While W is not empty
   a. Pick an element from W, call it X, and remove X from W
   b. For each successor Y of X in the graph such that DL[Y] = false do:
      i. Remove all X's from each string in Y.info
      ii. If λ is now in Y.info , then place Y into W and set DL[Y] to true.

Propagation begins by taking A from W A is then removed from each right-hand side found in S.info, reducing ABC to BC. S is not added to W since λ was not created. The algorithm continues with W = {D}. Processing D causes D to be erased from B.info and C.info λ is generated in C.info. DL[C] is set to true; C is placed into W . Processing C erases C from S .info, but is not generated. W is empty, so propagation terminates.

Diagram 3 shows the final disposition of the graph. Note that DL could be computed directly from the graph by producing a list of nodes whose information sets contain λ. In the algorithm above, DL was computed during propagation for the sake of efficiency.

Diagram 3




```
S -> ABC | CB
A -> a | lambda
B -> DE | b
C -> D | c
D -> d | lambda
E -> EB | e
```

X	DL[X]
S	False
A	True
B	False
C	True
D	True
E	False

Our second algorithm computes First(X), the set of terminals that can appear at the beginning of a string derived from a non-terminal X. Formally, First(X) = {a in T | X =>* aβ}. Note that we calculate a slightly different result from the classical First() function, which includes ε as an element of the result if X =>* λ. We consider our version simpler and use DL for that purpose.

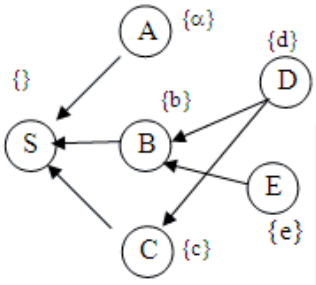
We setup the graph as follows:

1. For each X in N, create a node labeled X in the graph; set X.info = {}.
2. For each rule A → X_1 X_2 ... X_n (each X_i is in T+N)
   a. If X_1 in N add arc (X_1,A) to the graph; If X_1 in T add X_1 to A.info.
   b. Add arc (X_i,A) if DL[X_1] && DL[X_2] &&...&& DL[X_{i-1}] holds DL[t] = false for every t in T.

Diagram 4 shows the initial setup of the graph. Propagation proceeds as follows:

1. Initialize **W** to contain all nodes with non-empty information.
2. While W is not empty
   a. Remove an element **X** from W,
   b. For each Y such that arc (X,Y) is in the graph do
      i. Y.info = Y.info union X.info

Diagram 4




```
S -> ABC | CB
A -> a | lambda
B -> DE | b
C -> D | c
D -> d | lambda
E -> EB | e
```

X	DL[X]
S	False
A	True
B	False
C	True
D	True
E	False

ii. If Y.info changes, add Y to W

Propagation begins by initializing the working set W to all nodes which have non-empty information sets. All non-terminals are included, therefore, except for **S**. The initial information associated with a given node **A** are symbols which can appear first in a single-step derivation from **A**. The construction of the graph ensures the sets always contain symbols which can appear first in a derivation. When propagation completes, the sets contain all first symbols: **S:{a,b,c,d,e}; A:{a};B{b,d,e};C:{c,d}; D:{d}; E:{e}**

Our last example computes Follows(**X**), the set of terminals that follow a non-terminal in a string derived from S. Formally: Follows(**X**) = {**a** in T | **S** =>* αXa β}. To compute this we need to extend First to operate on strings of symbols. If α= $X_1X_2…X_n$ then First(α) = {a ιν T =>* a β}. This can be computed by First(X_1) U First(X_2) U … First(X_i) where i is the least value such that DL[Xi] is false.

To compute Follows using information flow we must determine when a terminal follows a non-terminal in a derivation. The trivial case is when a terminal follows a non-terminal in a grammar rule as `a` follows `X` in `Y → αXaβ`. However, when a grammar rule has a string of non-terminals, say X_1 X_2 ... X_n, then any member of the First($X_2…X_n$)will follow X_1.. If all the non-terminals X_2 ... X_n are *nullable* (i.e. each DL[X_i] is true), then whatever follows Y also follows each of X_1 ... X_n.

To implement these notions we initialize the graph as follows:

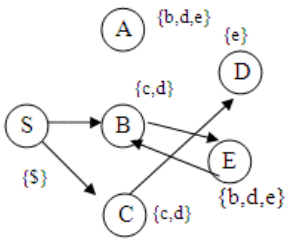
1. For each X in N, create a corresponding node labeled X in the graph.
2. For each grammar rule written in the form: Y → αXb, (X is a non-terminal),
   a. Include First(β) in X.info
   b. If β =>* λ include arc (A,X_i).

Consider the rule **S→ABC. C** follows `B,` so First`(C)` is added to `B.info`; since `B` follows `A,` First`(B)` is added to `A.info.` Because C terminates the rule, whatever follows S also follows C. (`S`, being the start symbol, is followed by the end-of-file token, $.) Because C is nullable, what follows S also follows `B.` Hence the graph contains arcs (S,B) and (S,C). Now consider `E→EB.` Since B terminates the rule, $ will follow B. B is added E.info; since B is not nullable, there is no further processing of the rule.

The propagation is identical to that for First. The final annotations are shown in Diagram 6.

Understanding how these graph algorithms work makes it easy to generate sample grammars for class or exam use.

Diagram 5

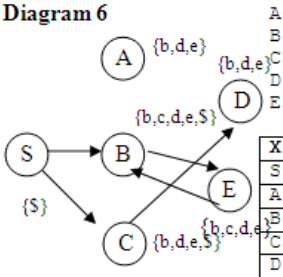



```
S -> ABC | CB
A -> a | lambda
B -> DE | b
C -> D | c
D -> d | lambda
E -> EB | e
```

X	DL[X]	First[X]
S	False	a,b,c,d,e
A	True	a
B	False	b,d,e
C	True	c,d
D	True	d
E	False	e

W = {A,B,C,D,E}

Diagram 6




```
S -> ABC | CB
A -> a | lambda
B -> DE | b
C -> D | c
D -> d | lambda
E -> EB | e
```

X	First[X]	Follows[X]
S	a,b,c,d,e	
A	a	b,d,e
B	b,d,e	b,c,d,e
C	c,d	
D	d	e
E	e	b,d,e

W = {A,B,C,D,E}

For example, consider a graph with 2 source nodes (A and B) feeding node D, which feeds node E which feeds the sink nodes (G and H). Initialize each node with at least one unique terminal. Now, consider ways in which students often misunderstand these algorithms. For example, for Follows( ), there will always be an arc from the left hand symbol to the final symbol in a rule, but only an arc to the second last symbol when the final symbol is nullable. (For the sake of brevity, we gloss over some minor details). Create a rule S → ABCDEFGH$, together with rules A→a ; B→b ; etc. to yield the unique terminals for seeding the algorithm. Rules like A→aD; and B→bD yield the arcs (A,D) and (B,D). The rule C→cDc ; should not yield the arc (C,D). D→E yields (D,E). Rules like E → FGH ; where H is nullable and G is not, yield arcs (E, H) and (E, G) but not (E,F).

CONCLUSION

Information flow analysis has been adapted to compute three standard analyses of grammars: DerivesLambda, First and Follows. This approach have proven advantageous in several ways: Because the algorithms have a common structure and can be readily visualized, less time is needed to teach this material. Students are able to learn the algorithms faster and to perform better when tested on this material. A pleasant benefit for the instructor is that this approach simplifies writing certain exam questions that have the right characteristics to check student understanding.

REFERENCES

[1] Fischer, C. and LeBlanc. *Crafting a Compiler with C*. Benjamin/Cummings, Redwood City, CA, 1991.

[2] Fosdick, Lloyd, and Leon J. Osterweil. “Data Flow Analysis in Software Reliability.” *ACM Computing Surveys 8,* 3 (Sept. 1976), 305-330.

[3] Aho, A.V.; ad Ullman, J.D. *The Theory of Parsing*, Translation and Compiling. Prentice-Hall, Englewood Cliffs, New Jersey, 1972.

STATE DRIVEN SEMANTIC MODELING OF OPERATORS IN ETL WORKFLOW*

Wesley Deneke, Wingning Li, and Craig Thompson
Computer Science and Computer Engineering Department
University of Arkansas
Fayetteville, AR 72701
479 575-6519
wdeneke@uark.edu

ABSTRACT

To process the flood of digital age data, ETL tools operating on grids have provided organizations with the ability to efficiently filter, clean, and persist very large data sets by means of complex workflows. Currently, however, constructing such workflows is largely manual, human time intensive, and error prone. Existing models omit the domain related knowledge necessary to validate the structure of such large, complex ETL workflows during construction. This paper introduces the concepts of preconditions, postconditions, and field abstraction on ETL operators to provide a richer model for ETL workflow that can leverage relevant domain knowledge to represent and enforce operator constraints.

INTRODUCTION

The term workflow can be defined as a set of ordered tasks to be performed to accomplish a goal. In Extract-Transform-Load (ETL) workflow, these tasks are operators. Operators can be generally described as having input fields and output fields. The input fields are a subset of columns from the dataset(s) in consideration that are mapped as arguments into the operator, where each input field denotes a specific dataset variable the operator will use during execution. Output fields denote modifications made to the current dataset(s) as a result of the operator's execution, which typically involves appending or updating specific fields (columns). More advanced operators may additionally include options, which are parameters used to alter the characteristics of an

operator's function. As many workflow operators are domain dependent, with the corresponding procedures written in languages like Java or C++, the underlying logic of these operators dictates conditions that must be true prior to execution in order to guarantee conditions after their execution. A valid workflow then is one that satisfies these conditions for each operator in the workflow. However, existing ETL tools neither sufficiently represent nor enforce these conditions. Instead, such conditions are currently captured only as knowledge in the heads of domain experts that construct ETL workflows. Users may unknowingly construct an invalid workflow and a violation may only be visible after execution when either the workflow fails to complete or incorrect results are produced. In domains that have hundreds of operators, each with possibly hundreds of input fields and options, ETL workflow specification becomes a human time intensive, error prone process. Organizations today are creating and collecting an overwhelming and ever increasing amount of data. ETL tools are a commonly adopted solution to aid in handling and processing this data. Generally, ETL tools employ operators sequenced in a user specified (defined) workflow to direct and manipulate data from source to destination. Sources can range from relational databases to flat files, with a typical destination being a data warehouse that can be used for, among other things, business intelligence and marketing campaigns. Between source and destination, transform operators that can filter, clean, and enhance the data are arranged in what can be a complex workflow to ensure a desired level of data quality is achieved before data integration [1].

ETL tools commonly model workflow as directed acyclic graphs, where nodes denote operators and data sources, edges represent the directed flow of data (inputs and outputs) between said nodes, and data fields are only represented by a name (an identifier) and primitive data type (int, varchar, double) [5]. While this is sufficient for users to visualize mapping data fields and ordering operators, this is insufficient with regard to workflow validation as it does not enforce any operator conditions besides ensuring fields of matching data type are mapped together. Constructing a valid workflow involves knowledge of field content and state. The inner logic of operators may expect certain field content beyond the data type to properly execute. For example, the input field firstName may expect "name" data and not accept "email" or "address" data, even though all of type varchar. Operators may also have a constrained set of valid input combinations. An operator may accept fields A, B, and C, but internally, a valid input may be A and B, or C. Finally, as it can be observed that operator transformations change the state of data, such as unparsed to parsed, unfiltered to filtered, etc, operators can require that the input data is in a specific state to execute properly. Data may need to be parsed before it can be filtered. This factor constrains the sequence that operators can be used in a workflow, pushing certain operators earlier in the workflow to satisfy the state dependence of later operators. As no current modeling approaches account for these constraints, workflow validation falls upon the user.

Several commercial ETL tools aid users in creating ETL workflows. IBM Datastage [8] is one such visual tool that allows users to select and drag desired operators into a work area and connect data fields between the operators to create a workflow. It comes prepackaged with several generic operators and additionally provides users the ability to incorporate custom C++ operators. These operators may have options and the software enforces that they are set and not left blank. Despite such features, this tool allows any

combination of fields to be mapped as an argument regardless of the content or state. As a result, operator conditions can be violated and invalid workflows may be unknowingly constructed by users.

While there has been little academic research on modeling ETL workflow, the paper Conceptual Modeling for ETL Processes [4] and subsequent works [5, 6] sought to add understanding of what must take place during loading a data warehouse via ETL processes and explain how the source attributes map to the destination. These works recognized the need to incorporate additional constraints on operators (referred to in this work as activities), where constraints could indicate an activity that should immediately precede the activity with said constraint. However, the proposed model does not recognize the content or state dependence of operators and workflow. Lacking the ability to represent and enforce operators conditions, this model would allow user to unknowingly construct invalid workflows.

The approach presented in this paper introduces new constructs to the model of ETL workflow, allowing operator constraints to be represented and enforced. Data types are abstracted into categories to ensure field content matches, attributed fields are used to represent field state to enforce proper sequencing of operators, and boolean expressions and predicates are used to represent valid input mappings. In this way, this model redirects the validation of a workflow's operator mapping and sequencing from the human user to the implementation of the model, ensuring only valid workflows can be built.

OBJECTIVE

The objective of this paper is to develop a richer model for ETL workflow that can leverage relevant domain knowledge to represent and enforce operator constraints by:

- Adding abstraction to data fields
- Describing and enforcing preconditions & postconditions of operators

ARCHITECTURE

The design presented by this paper is limited in scope by the following assumptions. First, as the focus of this research is transforming the data from source to destination, it is assumed that the data fields are properly identified. Thus, data identification is considered out of scope, as it can be handled by the related work of Layout Inferencing [3]. Next, the domain of operators under consideration will be limited to those with a defined set of input fields, output fields, and transformations.

With these assumptions in mind, this paper asserts the following model. Formally, each operator consists of a set of input fields, output fields, options, and an input requirement expression. Each input field and output field is defined by a name, data type, and an abstracted field category. The state of each field is represented by a set of attributes. An input requirement expression is a boolean expression, using descriptive predicates, representing what combination of fields (including attributes) is valid input. Options then consist of a set of possible value settings that are each represented by a set of output fields and an input requirement expression. The new constructs of field

categories, input requirement expressions, and attributes are explained in the following sections.

FIELD CATEGORY

Taking a lesson from abstraction concepts in object oriented analysis and design [10], different instances of a class can be related by object type. Similarly, data fields may have incongruent names, yet still be categorized (related) by content type. Abstracting data fields above their name and primitive type to a category can reflect knowledge about their contents to meet higher level semantic relationships or conditions. This model incorporates field categories as an abstraction of the content type in order to represent operator constraints on the content of input fields.

Consider a workflow where operator A is followed by operator B (ref Fig. 1A). If B's input field B1 expects "name" data and A's output field A3 contains "name" data, then operator B should be able to validate A3 as a viable input for B1 despite the name difference. Thus, it can be observed that data fields can also have object-like categorization. However, based on the domain a generic category such as "name" may not be sufficient to distinguish fields. A4 and B2 may also contain "name" data, but more specifically A4 and B2 may contain "last name" data while A3 and B1 contain "first name" data. Therefore, these categories could be extended to sub-categories, denoted as [A1::Name.First], [B2::Name.First], [B3::Name.Last], and [A2::Name.Last], for a finer grained semantic description of field content. Then each of these is related in that they inherited from the "Name" category, but they are also differentiable by their subcategory.

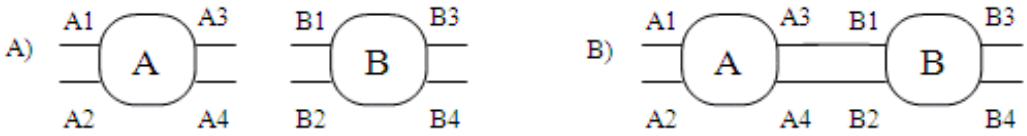


Figure 1

INPUT REQUIREMENT EXPRESSION

An input requirement expression is used to represent an operator's precondition. In programming languages, a precondition is a condition that must be true prior to a program's execution in order to guarantee the condition after execution, known as the postcondition [7,9]. This model incorporates these concepts into ETL workflow, representing the operator constraints on the content, state, and combination of input fields as a boolean expression. Specifically, the input requirement expression represents the subset of data an operator can accept as input and have a valid execution. Taking this idea a step further, predicates [11] may also be incorporated into the input requirement expression. Predicates are descriptive modifiers to input requirements, such as required, optional, or specifically disallowed, providing semantic content to describing acceptable input. To illustrate these ideas, consider the following example using Figure 1. Assume operator A's precondition requires either an email address (input A1) or a phone

number (input A2), but can only take one field as input or operator A will error. Operator A's input requirement expression (precondition) would then be represented as ([A1::Name.First].Required && [A2::Name.Last].NA) || ([A1::Name.First].NA && [A2::Name.Last].Required), where the ".Required" predicate evaluates to true only if the input it modifies is mapped in and ".NA" evaluates to true only if the input it modifies is not mapped in.

ATTRIBUTED FIELDS

Operator's transform data between states. Data enters an operator in one state and exits in a new state. Attributes are semantic descriptors that can be appended to fields post-execution to represent relevant knowledge about field state [2]. This post-execution state can then be represented by the output fields of the operator with relevant attributes appended to them; this would be the operator's postcondition. These attributes may then also be incorporated into each operator's input requirement expression to represent preconditions on the state of input fields. In this way, the sequencing of operators to attain a valid workflow may be driven by the state of the data.

Assume operator A's output fields A3 and A4 (ref Fig 1A) have the postcondition "corrected". This can be represented using attributes as [A3::Name.First + Corrected] and [A4::Name.Last + Corrected]. Also assume operator B's output fields have the postcondition "validated", represented as [B3::Name.First + Validated] and [B4::Name.Last + Validated], and B's input fields have the precondition "corrected", represented as [B1::Name.First + Corrected] and [B2::Name.Last + Corrected]. Thus, in constructing a workflow involving these two operators, the operator A would have to be sequenced some time before operator B to satisfy B's preconditions, otherwise the workflow could be invalid.

OPTIONS

The concepts of preconditions and postconditions may also be applied to an operator's options to capture how they affect operator conditions. While the options may not add input fields to an operator, an option's setting may affect the state and combination of input fields that are valid. This option precondition is represented by the option setting's input requirement expression. Also, as an option's setting may also affect the output state of data from an operator, each option setting has a set of output fields (including attributes) to represent the option's postcondition. Performing a logical AND of the operator's and each option's input requirement expressions would represent the operator's complete precondition. If an input mapping was valid for this precondition, then both the operator's and option's postconditions would be applied.

Consider an option, B-Op1, on operator B that can have a value Y or N. The N setting has no pre or post conditions, but the Y setting has the precondition [B1::Name.First].Required and the postcondition [B3::Name.First + B-Op1]. Assuming that operator B's precondition is

[B1::Name.First + Corrected].Required || [B2::Name.Last + Corrected].Required and taking B's postcondition from the previous example, if B-Op1 is set to Y then a first name

that is corrected must always be mapped in to satisfy both the operator and option precondition. Then the resultant postcondition of B by combining both the operator and option postconditions would be [B3::Name.First + Validated + B-Op1].

CONCLUSIONS

The approach presented in this paper has introduced how to use field abstraction, attributes, and input requirement expressions in order to represent and enforce operator constraints. Adding an abstracted category to each field allows for homogeneity between fields with related content, clarifying relationships between the inputs and outputs of different operators. Appending domain-specific attributes to fields as semantic descriptors of data state provides a means of representing state related preconditions and postconditions. Input requirement expressions then allow these preconditions to be enforced on both the operators and options. By capturing the state and content related domain knowledge of operator conditions in this model, validation of a workflow's operator mapping and sequencing may be handled implicitly during construction rather than being stipulated only in the mind of the workflow designer.

FUTURE WORK

Looking forward, a use case for this model would be an implementation as an overlay for an existing workflow domain to perform workflow validation. Model constructs (fields, input requirement expressions, options) for each operator could be encoded in an object-oriented or relational fashion. Logic can then test the pre and post conditions during construction to validate or invalidate a workflow. Another natural next step would be to incorporate workflow standards and preference related domain knowledge with a workflow engine (i.e., an AI planner) to automatically generate a workflow to achieve a given goal state. Taking advantage of these ideas, our present work is on abstracting the specification of the workflow's end goal using a higher level intent language. Additional research paths include using attributes to determine the heritage of data fields through a workflow, deriving a proof for workflow correctness for a given workflow intent from operator preconditions and postconditions, and exploring optimization based on cost or execution time.

REFERENCES

[1] Deneke, W., J. Eno, W. Li, C. Thompson, Towards a Domain-Specific Modeling Language for Customer Data Integration Workflow, *Grid and Pervasive Computing Workshop*, Kunming, China, May 2008.

[2] Eno, J., C. Thompson, W. Li, W. Deneke., Enhanced Workflow Service Modeling, Conference on Applied Research in Information Technology, Acxiom Laboratory for Applied Research, February, Conway AR, 2008.

[3] Phillips, R., W. Li, G. Beavers, C. Thompson, J. Loghry, D. Nash, Layout Inference: Using Content Type Domain to Infer Record Structure, Conference on

Applied Research in Information Technology, Acxiom Laboratory for Applied Research, Conway AR, February 2009.

[4] Vassiliadis, P., A. Simitsis, S. Skiadopoulos, Conceptual Modeling for ETL Processes, ACM International Workshop on Data warehousing and OLAP (DOLAP), McLean VA, 2002.

[5] Simitsis, A., P. Vassiliadis, T. Sellis, Optimizing ETL Processes in Data Warehouses, 21st Intl. Conference on Data Engineering (ICDE), Tokyo, Japan, April 2005.

[6] Simitsis, A., Mapping Conceptual to Logical Models for ETL Processes, ACM International Workshop on Data warehousing and OLAP (DOLAP), Bremen, Germany, 2005.

[7] Fikes, R., Nilsson, N., STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence*, 2, (3-4), 189-208, 1971.

[8] http://www-01.ibm.com/software/data/infosphere/datastage/

[9] Gries, D., *The Science of Programming*, New York, NY: Springer-Verlag, 1981.

[10] Booch, G., *Object-oriented Analysis and Design with Applications* (3rd Ed.), Upper Saddle River, NJ: Addison-Wesley 2007.

[11] Barwise, J., An introduction to first-order logic, *Handbook of Mathematical Logic*, Chapter A.1, 6-47. Amsetdam, The Netherlands: Elsevier 1977.

A BOOT CAMP APPROACH TO LEARNING PROGRAMMING IN A CS0 COURSE*

John Stamey, Steve Sheel
{jwstamey | steves} @ coastal.edu
Coastal Carolina University, Conway, SC
jwstamey@coastal.edu, steves@coastal.edu

ABSTRACT

This research chronicles the experience a computer science department that switched to a "boot camp" approach in the CS0 programming lab course. This curriculum approach required students to write over 350 short programs during the semester. The percentage of students who were successful in completing the CS0 lab course dramatically improved during the two semesters in which the boot camp approach was used.

INTRODUCTION

Back in college, we all took calculus. If we were successful, it was more than likely we had a lecture followed by an assigned a series of 20 (or so) problems to work on between the lecture and the next class. Assuming we wanted to do well in the course, we were most likely diligent in doing our homework problems. At least, that's the way it was twenty years ago.

In 2009, there are challenges in teaching introductory programming to students. These (typically) 18 and 19 year olds are classified as millennials [1], in general born after 1985. They have an information-age mindset [2] that "Doing is more important than knowing. Knowledge is no longer perceived to be the ultimate goal, particularly in light of the fact that the half-life if information is so short. Results and actions are considered more important than the accumulation of facts." This leads us to understand why a class or lab filled with active work would be more appealing to millennials than a traditional lecture situation.

BACKGROUND

In response to the millennial mindset, a curriculum has been developed that focuses on important introductory computing concepts reinforced by many short programs to be written directly during a lab session. This "boot camp" approach has been used to teach a weekly, introductory (CS0) programming lab at a mid-sized university in the southeast. The curriculum is based on a set of twenty sets of programming, containing a total of 350 short programs to be written by the students and reviewed by the end of each 75-minute lab session. The labs were written by a member of the faculty who has over ten years experience in teaching programming labs at the CS0-CS2 level, as well as many years of computer programming practice and experience.

The idea of many short programs (similar to the idea of a "boot camp") involves repetition and Learning By Doing. Bromage and Mayer [3] found positive effects of repetition. The amount of correctly recalled information increased with repetition; structurally important information (such as the main idea of a paragraph) was remembered better with increased with repetition. Most important to programming, functionally important information (such as names of components) was remembered better than unimportant information with increased with repetition. Winslow [4] found undergraduates became successful in programming with strong, persistent practice. The Computer Science Boot Camp approach features over 350 programs that are to be written during twenty programming lab sessions.

Anzai and Simon [5], in their approach Learning By Doing, concluded that dividing a problem into smaller and smaller parts helped in completing a larger problem. Their problem domain for their study was the Towers of Hanoi - a classic problem in both recursion and programming. Pirollo and Anderson [6] found similar results when students translated written solutions to recursive programs to the computer. Repetitive exercise sessions were also found to be more useful in learning than traditional lectures for beginning (novice) programmers. [7] The National Clearinghouse on Families and Youth reports that learning by doing is an excellent approach to connect students with technology to help build skills that will translate into job opportunities later down the road. [8]

The Computer Science Boot Camp approach of teaching sub-goals to achieve a goal is similar. We would teach the IF statement and then frame it into the programming solution of a problem such "Is the length of a phrase greater than 10?" We note that Learning By Doing is at Level 3 of Bloom's Revised Taxonomy, "Apply." [9]

The CS0 lab had been taught for three semesters, from Spring 2007 to Fall 2008 in conjunction with the CS0 classroom course. Students received three credits for the course and one credit for the lab. During the three semesters in which a lab was taught as part of the CS0 course. A "passing" grade of C or better the CS0 lab meant was eligible to take CS1. In the first three semesters, the average passing rate was just over 76%. Feeling the need to improve the passing percentage out of CS0 lab, the Boot Camp approach was implemented in Spring 2009.

The language selected for the lab exercises was Python (currently 3.1), which available free of charge and downloadable from `www.python.org`. Python has a

positive reputation for being used to teach basic programming concepts and constructs. [10] These include:

- Python operators, if statements and loop constructs are useful in subsequent programming languages
- Python has excellent string processing and regular expression functions/methods
- Python supports arrays, functions and modules.

The Python development environment, IDLE, helps the instructor to see all the work that has been done in a lab by scrolling through the output found in one display window.

CURRICULUM TOPICS

In the labs and on the tests, there is an emphasis on the three constructs of structured programming. Programming skills stressed in the curriculum included:

Statements in Sequence

- Writing over 100 programs to solve everyday problems involving basic mathematics, statistics, along with string input and handling

Iteration of Statements

- Writing FOR loops and WHILE loops (counting loops)
- Converting FOR loops to WHILE loops, and WHILE loops to FOR loops
- Using WHILE loops to collect input to compute sums, averages and standard deviations

Selection of Statements

- Write short programs requiring solutions with IF and IF-ELSE statements
- Using IF statements to perform validation of input (such as emails, phone numbers and zip codes –traditional topics covered in secure software engineering)

Other Topics

- Reading and writing CSV files
- Creating user-defined functions
- Program tracing

LIMITS OF THE CS0 CURRICULUM

An important program for the students to master is writing a WHILE loop to enter test scores, computing the average of the test scores once a sentinel value (any negative number) is entered. The program, found in Figure 1, requires the following seven key concepts/sub-goals:

Initialize variables

- A priming read, with casting
- A WHILE loop with condition
- Accumulating totals
- IF statement to avoid division by zero
- Arithmetic computations (`count, sum, score, average`)

We note that seven is the approximate number of things one can keep in short term memory. [11] For this reason, a program of this complexity – with seven key concepts/sub-goals, is the program of maximum difficulty for the Computer Science Boot Camp.

```
def main():
   sum = 0
   count = 0
   score = int(input("Enter a test score, negative number to stop")
   while(score >= 0):
      count = count + 1
      sum = sum + score
      score = int(input("Enter a test score, negative number to stop")
   if(count = 0):
       print("No scores were entered")
   else:
      average = sum / count
      print("The average of ", count, "scores was ", average)
main()
```

Figure 1

RESULTS OF THE BOOT CAMP APPROACH

The majority of the test questions dealt with writing code to specifications (70%). The remaining questions dealt with concepts and tracing programs. Statistics on tests across the seven sections for Fall 2009 are found in *Table 1*.

Test	n	Mean	Standard Deviation	Range
1	81	84.80	15.80	[100, 52]
2	79	87.46	12.12	[100, 51]
3	78	85.52	12.04	[100, 49]

Table 1: Results of three in-class tests

The department was very pleased with the average scores on the tests. Due to the code-intensive questions, we feel the students are learning the important core skills set out in the Course Objectives.

During Spring and Fall 2009, the passing rate rose dramatically, averaging close to 94%, a statistically significant difference at α=.05. The data may be found in *Table 2*.

Semester	Students Taking CS0 lab	Students Passing CS0 lab	Passing Percentage
Fall 2007 *	71	54	76.06%
Spring 2008 *	34	26	76.47%
Fall 2008 *	92	70	76.09%
Spring 2009 **	42	40	95.24%
Fall 2009 **	92	85	92.39%

Table 2: Results of Students Passing CS0 lab:
** = performance prior to Boot Camp curriculum;*
*** = performances with Boot Camp curriculum*

ADVANTAGES OF THE APPROACH AND STUDENT FEEDBACK

Advantages are numerous when using the "boot camp" approach in the lab. The pre-written labs to the instructor allowed for:

- Minimum preparation time for the instructor
- Ease of conducting a class - a short talk, followed by helping the students in place
- Ease of making sure the programs are done (with the Python IDLE)

Advantages of the labs to the student

- Repetition and Learning By Doing
- Short programs use subgoals (programming constructs) to provide immediate solutions to real-world problems
- Students can usually work until completion, thus using their 75 minute lab learning and doing work

A mid-term survey was given to n=77 students with the following four questions:

- Q_1: Using Python has been a positive experience
- Q_2: The use of many short programs has been helpful in learning
- Q_3: The instructions in the lab materials are generally straightforward and easy to understand.
- Q_4: I feel I am making positive progress toward learning programming

Descriptive statistics and confidence intervals, summarized in *Table 3*, show the students felt they were having a very positive experience from the choice of the language, and the multitude of short programming exercises.

Question	Mean	Std. Dev.	95% Confidence Interval
1	4.276	0.778	(2.911, 3.089)
2	4.414	0.773	(2.912, 3.088)
3	4.224	0.839	(2.904, 3.096)
4	4.534	0.754	(2.914, 3.086)

Table 3: Descriptive statistics and 95% confidence interval from student opinion survey

Based on comments on the survey, many students seem to be pleased with their progress and feel their time is being well-spent in the programming-intensive labs.

CONCLUSIONS

The "boot camp" format of the CS0 lab material [12] proved to be successful through its two semester of use. Students and faculty are on-board with the concept and the Department is looking forward to improved performance in CS1 (and above) courses from these students, based on their mastery of the concepts of structured programming in the CS0 lab. Further studies will continue as the department measures and tracks the performance of this group of students through the CS1 and CS2 courses.

REFERENCES

[1] Oblinger, D., Boomers, Gen-Xers and Millennials: Understanding the New Students, *EDUCAUSE Review Magazine*, 38, (4), 2003.

[2] Frand, J.L., The Information-Age Mindset: Changes in Students and Implications for Higher Education, *EDUCAUSE Review Magazine*, 35, (5), 2000.

[3] Bromage, B.K., Mayer, R. E., Quantitative and qualitative effects of repetition on learning from technical text, *Journal of Educational Psychology*, 78, (4), 271-278, 1986.

[4] Winslow, L.E. Programming Pedagogy – A Psychological Overview, *SIGCSE Bulletin*, 28, (3), 17-25, September 1996.

[5] Anzai, Y., Simon, H.A., The Theory of Learning by Doing. *Psychological Review*, 86, (2), 124-140, 1979.

[6] Pirolli, P.L. Anderson, J.R. The role of learning from examples in the acquisition of recursive programming skills, *Canadian Journal of Psychology*, 39, (2), 240-272, June 1985.

[7] Lahitnen, E., Ala-Mutka, K. & Jarvinen, H. A Study of the Difficulties of Novice Programmers. *Proceedings of ACM SIGCSE ITiCSE '05*, Monte de Caparica, Portugal, June 27-9, 2005.

[8] National Clearinghouse on Families & Youth, Learning By Doing: Connecting with Youth through Technology, n.d., ncfy.acf.hhs.gov/publications/lbd/tech.htm/, retrieved December 1, 2009.

[9] Anderson, L.W., Krathwol, D.R. (Eds.), *A taxonomy for learning, teaching and assessing: revision of Bloom's Taxonomy of educational outcomes*, New York: Longman, 2001.

[10] Oldham, J.D. What Happens after Python in CS1? *Journal of Computing Sciences in Colleges*, 20, (6), 7-13, 2005.

[11] Miller, G.A., The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information, *The Psychological Review*, 63, (2), 81-89, 1956.

[12] Stamey, J.W., *Computer Science Boot Camp, Featuring Python 3.1*, Myrtle Beach, SC: SoftwareEngineeringOnline.com, 2009.

A PROGRAMMING REMEDIATION PLAN*

Andrea Edwards, Kun Zhang
Xavier University of Louisiana
Department of Computer Science
1 Drexel Drive
New Orleans, Louisiana 70125 USA
504.520.7336
aedwards@xula.edu, kzhang@xula.edu

ABSTRACT

Many of the difficulties in junior and senior level computing courses stem from deficiencies in programming skills. This paper describes a remediation tool (PREP) designed for computing students who have completed the introductory programming sequence but have an atrophy of programming skills and knowledge. The strength of this self-paced system is based on a compilation of programming problems with solution hints, a set of similar problems for cognitive recognition, and simulations that encourage students to explore complex problem sets. PREP has been successful in improving student learning, especially those students in need of programming remediation.

INTRODUCTION

Computing is an important area of study with engaging topics about emerging technologies; a global need for the advances brought to technology-available communities; as well as seemingly unlimited opportunities for new development, stable work, and good salaries. Yet, the computing crisis persists. The crisis is well documented having a drastic decline in enrollments in the past decade, fewer college students expressing interests in computing studies, a smaller number of good graduates, yet an increasing world-wide need for high-level computing knowledge and skills [7]. This paper considers ways of improving the foundational background of current computing students. The goal is to provide opportunities for "eureka moments" to students so they are (1) self-motivated to engage the difficult content in junior and senior

computing courses; (2) have a solid basis for completing those courses; and (3) have a proper assessment of their capabilities in this challenging field of study.

All people need *eureka moments*: the moments of satisfaction after a discovery. These moments are individually attained and cannot be given or taken by another. Some of today's computing students are missing these intermediate cries of joy. It is not that they are unsuccessful learners; instead they seem to be lacking the joy of self-discovery. The content in the junior and senior computing courses is what attracted most students to computing studies but that content is far a field of the introductory course content in the freshman and sophomore computing courses. Many students begin computing curricula without the stamina to persist through the foundational material.

Programming is a foundational skill in a computing education. Students who are proficient programmers have a basis for more advanced computing concepts while under-prepared programmers are dramatically hindered by their programming deficiencies. Retention is affected because students who have difficulties completing the introductory programming sequence change their area of study. Some students earn minimal passing credit in the introductory programming sequence and continue to struggle through subsequent course content while other students earn passing credit in the introductory programming sequence but their programming skills seem to atrophy in subsequent semesters. This paper addresses these students who are in need of remediation.

Most introductory programming sequences include one or two basic programming courses (listed as CS I in ACM's Computing Curricula) followed by a data structures course (CS II). CS I requires knowledge of primitive data types, control structures, functions, a collection container (for example an array), and usually a user-defined data type. Data Structures (CS II) is the gatekeeper course to the advanced computer science courses. It is CS II that teaches the implementation strategies and structures most useful for upper level computing courses. This paper is most interested in the students who successfully complete CS II but seem unable to apply those strategies and structures once in upper level courses. *These students are identified as **remedial** because they have successfully completed CS II but have programming deficiencies in subsequent courses*. This paper has effective techniques of intense tutoring designed to help students improve their programming knowledge and skills. In accord with [5], these tutoring techniques are most effective for less proficient students who are reviewing material rather than for students being introduced to the material. The purpose of this supplemental remedial instruction is to assist students to overcome their programming deficiencies so they may be successful in advanced computer science courses.

Intelligent tutoring systems are not new. Tutoring systems like Carnegie Learning's Cognitive Tutor [6] and the Andes Tutoring System [8] give individualized instruction and make inferences about student work using an artificial intelligence component while supplementing student learning through step-by-step feedback. The instructional software of this paper uses human expertise rather than artificial intelligence and provides human-expert programmer direction to student solutions rather than revealing actual solutions. This paper discusses software that uses tutored problem solving to remediate programming skills.

Remediation is required for students who lack the academic skills needed to be successful in college courses. In the United States, mathematics and writing programs

have a history of remedial courses. Although remediation is not a new practice, some institutions are unwilling (or unable) to provide remedial (non-college) level work at the college level [1, 3]. The purpose of this paper is not to join the discussions about remediation of under prepared students; instead this paper is about students who, once prepared as programmers, are no longer ready to be successful programmers. If for no other reason, the computing crisis makes programming remediation a worthy endeavor.

A PROGRAMMING REMEDIATION PLAN (PREP)

Typical college-level remediation courses are not-for-credit but have the cost and time demands of a credit-bearing course. The programming remediation plan in this paper is also not for credit. It is extracurricular with a continuum of helpful interventions ranging from identification of programming deficiencies, individualized programming assignments, faculty tutors, computer-based instruction, and consequences if the student fails to demonstrate improvement. Participation in the plan extends a student's class day but no college credit is earned (since the student has already successfully completed the courses in the introductory programming sequence.) The data shows that students motivated to complete their programming remediation plan are better prepared for subsequent computing courses --- this remediation plan is effective.

The PREP system has three major components: (1) A large number of computer-based learning modules especially those covering the CS I content; (2) Students in need of remediation, motivated to participate, or interested in improving their programming; and (3) Faculty, those most familiar with the programming sequence, must be willing to encourage and sometimes stipulate use of the system.

PREP began as merely a test bank of programming questions. During the class immediately prior to a test, students often ask for "sample" test questions. This compilation of test questions was usually offered with little context but with solution programs. The students eagerly anticipated these questions but seemed unsure of how to best study the material. They would often read the solution programs even before reading the sample test questions. Because computing is much more about *doing* than *reading*, these sample test questions seemed to be of little help from the instructor's perspective but allayed student's curiosity about the test.

As the test bank grew, it became convenient to categorize each question by content and by level of difficulty. The content categories follow the typical imperative programming paradigm of sequential, selection, repetition, arrays, functions, and classes. The levels of difficulty are introductory, intermediate, or advanced categories. The test bank had a partial order but was still more about reading rather than doing programming.

Finally, the test bank of programming questions was integrated into a technology enhanced learning system. This web application is available to all students at any time and is self-paced. The system was totally in-house using Linux, Apache, MySQL, PHP, and Ruby on Rails. Students login the system using their student identification number then either begin with the last problem solved or have the option of changing to another problem, category, or level of difficulty. Figure 1 is a condensed screenshot of PREP's initial screen after login. The system displays the student's last login timestamp and the last problem reviewed. This is the main screen. Figure 2 is a screenshot of a

programming problem showing how the user can show a solution hint, review similar problems, show a simulation of the problem, or can return to the main screen.

The strength of this system is based on the functionality shown in Figure 2. A student has options to get help via a hint, to review similar problems, and to see simulations of ways the problem is used in more complex programs. Because students have some familiarity with these categories of problems (having completed CS II), the goal is to help them recognize similar problems and encourage curiosity by displaying simulations.

An example may be useful. Suppose the student wants to review introductory level loops via the problem "Write a program to output the integers from 1 to 10." A student who does not need assistance with this problem could open their favorite developer environment and solve the problem. A student a bit unsure of a solution can read the list of similar problems and jog their memory on how to begin or they could show a hint." The hint for this particular problem is "Your code should output 1 then increment a counter, output 2 then increment again, output 3 then increment, … , until you have output 10 integers in sequence." The hints are intended to be sufficient for guiding the student verbally through a solution. The hints are written as if an expert programmer were described the problem, then describing in more depth, and further hints with deeper explanation of the problem solution. The simulations are to encourage curiosity about more advanced programming. The simulation for this particular problem is a digital clock that waits ten ticks between time states. At any time the student can decide to continue their review of similar problems or begin a different set of problems.

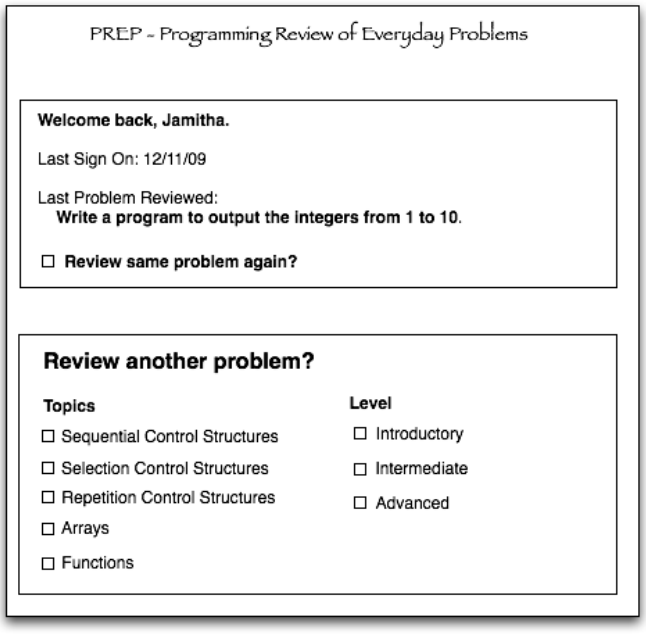


Figure #1 – Condensed PREP Screenshot

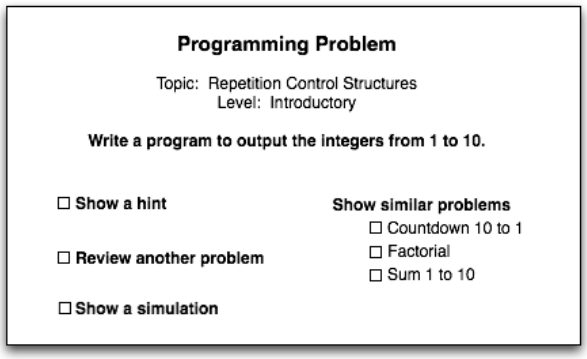


Figure #2 –PREP Programming Problem Screenshot

PREP is a remediation system that is self-paced, interesting, and provides sufficient guidance for students to solve problems with minimal assistance from others. Students who are uncomfortable asking for help with material previously covered will be able to practice solving problems in the context of their greatest need. Students interested in a refresher can solve problems in sequence. Other students may enjoy the challenge of considering solutions to the simulations.

Two aspects of the system are supervised: (1) the initial assessment of the student's programming abilities and (2) assessing the quality of student solutions to the review problems. The system is most effective after a comprehensive assessment of a student's programming proficiencies. This assessment is a short (less than an hour) pencil-and-paper problem set that surveys each topic and is weighted toward student's understanding of selection, repetition, and arrays. All faculty members can easily score the assessment problem set using the provided rubric. What is more difficult and time-consuming is scoring the solution problems that are not completed as course assignments. Good programming is more than just producing the correct output so instructor comments on the student solutions must be as thorough as in any introductory programming course. This can result in extraordinary work for the faculty member but is time-critical in retaining student interest in improvement and provides necessary feedback for their self-reflection.

All students have access to PREP (a web application but behind the university's firewall) and assessments are scored and reported to students as soon as possible and students in introductory programming courses receive feedback within 24 hours. With that, the students are instructed to allot at least one hour to each remediation session, have pencil and paper for independent solutions, and are assigned a faculty programming coach with responsibilities for immediate scoring of student solutions.

THE RESULTS

PREP is widely used by students in the introductory computing courses. Although intended to remediate students who have successfully completed CS II, it is most used by students in the CS I courses. CS II students only use PREP when required by their instructor in subsequent courses. It seems that their programming deficiencies must be pinpointed before they are willing to consider remediation. Conversely, the beginning programmers in the CS I courses solve PREP problems sometimes taking time away from their course assignments. They seem challenged by progressing through the difficulty levels. One cohort of beginning programmers even posted progress reports listing the problems they were able to solve. Solving PREP problems has become a competition with teams of beginning programmers working together to discuss and design solutions to the simulations without faculty leadership. Surprisingly, even non-computing students have joined the competitions. It may be that PREP is a way to retain and also recruit good students to computing.

Students use PREP during regular school hours but the greatest collaborative work occurs in the early evenings and on weekends. This is a welcome addition since remedial students sparsely attend course-based review sessions especially during evening and weekend sessions. Eight-five percent of the computing students have accessed PREP and submitted at least one solution that is not required by a computing course.

The strengths of PREP include the following: (1) students are engaged in voluntary programming so are improving their proficiency and understanding; (2) faculty are dedicated to providing specific feedback and encouraging student motivation, rapport, and high expectations; (3) peer instruction and small group collaborations have resulted in creative, hands-on problem solving and learning activities; (4) individual student weaknesses are addressed with differentiated instruction so students ask directed questions based on their specific needs; and (5) the PREP system is a comfortable learning environment and students enjoy recommending hints, similar problems, and even simulations.

Most of these results are anecdotal but of the twenty students who have used PREP for remediation, their scores on programming assignments have improved, they are more conversant about programming, they have a better understanding of their proficiency, and are they seem more willing to ask for help in their current computing courses.

CONCLUSION

Of course, remediation occurs in many ways and on many different levels. High quality instruction has the greatest impact on student achievement but when a student's skills erode, PREP has been successful in improving student learning, especially those students in need of programming remediation.

REFERENCES

[1] Bettinger, E., Long T. B., Does College Remediation Work?, *The Journal of Human Resources*, 44, (3), 736-771, 2009.

[2] Furst, M., Isbell, C., Guzdial, M., THREADS: How to restructure a computer science curriculum for a flat world, *Proceedings of the Thirty-Eighth SIGCSE Technical Symposium on Computer Science Education*, 420-424, 2007.

[3] Kentucky Council on Post-Secondary Education, Fall 2007, http://cpe.ky.gov/info/dev_edu/, retrieved December 10, 2009.

[4] Pedroni, M., Meyer, B, The Inverted Curriculum in Practice, *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, 481-485, 2006.

[5] Rose, M., Colleges need to re-Mediate Remediation, *The Chronicle of Higher Education*, 8/3/2009.

[6] Razzaq, L., Hefferman, N., Tutor or Not to Tutor: That is the Question, *Frontiers in Artificial Intelligence and Applications*, 457-464, 2009.

[7] Ritter, S., et al., Cognitive Tutor: Applied research in mathematics education, *Psychonomic Bulletin & Review*, 14, (2), 249-255, 2007.

[8] Shackelford, Russ (chair) et al., Computing Curricula 2005: The Overview Report, *Computing Curricula series produced by the Joint Task Force for Computing Curricula 2005*, 2006.

[9] Vaniehn, Kurt et al., The Andes Physics Tutoring System: Lessons Learned, *International Journal of Artificial Intelligence in Education*, 15, (3), 147-204, 2005.

A GRAPHICAL FRAMEWORK FOR ASSISTING PROOFS*

Cong-Cong Xing
Department of Mathematics and Computer Science
Nicholls State University
Thibodaux, LA 70310
cong-cong.xing@nicholls.edu

ABSTRACT

Theories in computer science, especially proofs, are a tough topic for the student. Most students feel great bewilderment when studying example proofs and writing their own proofs. In this short paper, we first identify some common problems that students face when dealing with proofs, then define a graphical framework for facilitating the reading and writing of proofs, next present some proof examples under this framework, and finally summarize what can be gained by this framework.

1 INTRODUCTION

It is well-known that theory courses in computer science, such as Discrete Math, Analysis of Algorithms, and Theory of Computation, are of fundamental importance in computer science education (see e.g. [6, 8, 7]). Unfortunately, as we have experienced, theory courses are more difficult to learn for students and more challenging to teach for instructors than non-theory computer science courses. In particular, (math) proofs associated with various topics in theory courses are a substantial barrier. Students generally feel that proofs found in textbooks are difficult to follow and understand, and do not know how to proceed when writing their own proofs. Among all possible factors, we believe that the two primary reasons that cause this situation are as follows.

- *Student math background.* In our institution, the average ACT math scores of computer science majors is only 20 (or equivalently, 56 of out 100). Students with such a background are not sufficiently prepared to handle college level math and theory courses, and not surprisingly, are having a difficult time in dealing with proofs.

- *The way in which proofs are presented in textbooks.* The usual way to present proofs, as found in most textbooks, is to use the combination of regular English statements with math notations. In this "linear" fashion of proofs, the logical reasoning process and structure, which is critical for students to understand proofs, are embedded in the lines of proofs, and may not appear to be explicitly clear to the student. Students, when reading this kind of proofs, may feel unguided in terms of how to write such a proof on their own.

While improving students' math background does not seem to be feasible in a short period of time (one or two semesters), we tackle the latter by proposing a graphical framework for conducting proofs. This framework allows the reasoning process and structure to be visualized clearly.

2 THE PROBLEM

Students show various kinds of troubles when working with proofs. Due to the page limit, here, we illustrate some typical troubles using two short examples.

Example 1 Traditional proofs are written in the format of plain English sentences combined with math notations. For instance, the following proof for $A - (B \cup C) \subseteq (A - B) \cap (A - C)$ is taken from [9].

> Let x be any element of $A - (B \cup C)$; then $x \in A$, but $x \notin B$ and $x \notin C$. Hence, x is an element of both $A - B$ and $A - C$, and is therefore an element of $A - (B \cup C)$.

There are only two sentences in this proof. However, students, when reading such a short proof,may have the following problems.

(1) They do not know that the word "but" in the first sentence actually means "and".

(2) It is not clear to them why $x \notin B$ and $x \notin C$ are true. In particular, why $x \notin B$ and $x \notin C$ are connected by "and", not by "or".

(3) Due to (1) and (2), students are not sure about what makes true "... x is an element of both $A - B$ and $A - C$...".

Example 2 Suppose students are given this problem: show $n^2 + n + 1 \geq 3n$ for all positive integer n. Many of them would write something like the following as a proof.

$$n^2 + n + 1 \geq 3n$$
$$\Downarrow$$
$$n^2 + 1 \geq 2n$$
$$\Downarrow$$
$$n^2 \geq 2n - 1$$
$$\Downarrow$$
$$n \geq 2 - 1/n$$

When being asked to explain this proof, students' response would be: take the first line, subtracting n from both sides yields the second line; subtracting 1 from both sides of the second line gives the third line; and dividing n on both sides of the third line yields the last line. Since the last line is true when $n \geq 2$, the proof is finished.

This kind of argument is deceptive as each line is generated by a sound math operation. There are two things that are worthy of noting: (1) Students are making a fundamental and fatal error here.they have assumed $n^2 + n + 1 \geq 3n$ (which is actually what to be proved) is true from the beginning, and all subsequent inferences are based on that. (2) Besides, what is really proved by the above lines is "$n^2 + n + 1 \geq 3n$ implies $n \geq 2 - 1/n$", not "$n^2 + n + 1 \geq 3$" at all.

3 THE FRAMEWORK

Toward resolving problems as illustrated in the previous section, we, in this section, propose a graphical framework for proofs. This framework makes the logical reasoning process and structure explicitly clear by blending them into the syntax, and thereby provides a general guidance for students in the sense of learning and conducting proofs.

3.1 The Notation

Under this framework, a proof is essentially a directed graph specialized for logical reasoning purpose. Details are as follows:

- The graph is layered into multiple levels (or stages) where (sub-)facts and/or (sub-)goals are displayed. The top level is formed by displaying what is originally given/known/assumed. Each subsequent level is formed by inference (or "reverse inference") from (as many as necessary) previous facts and/or goals.
- Nodes and edges can be either solid (__) or dotted (...). Solid nodes and edges represent what is occurring in the current proof process whereas dotted nodes and edges represent what will occur in the future proof process or has occurred in a previous proof process.
- A rectangle node represents a fact which is either given, or known, or assumed, or derived in the process of reasoning.
- An oval node represents the goal (to be proved) or a subgoal in the process of reasoning.
- An edge indicates an inference. Namely, what is represented by the target node of an edge is derived from what is represented by the source node of the edge.
- An edge is labeled by what causes the inference.
- Multiple edges may be coupled to indicate an inference. This case is indicated by an arc connecting these edges, and a single labeling is used for the these edges.

3.2 The Reasoning Process

The logical reasoning process and structure, by which a proof is shaped under this framework, can be characterized as top-down, bottom-up, and hybrid. In each of these approaches, the facts are always put at the top of the graph, and the goal is always put at the bottom of the graph. A proof, thus, amounts to completing the graph by filling out what is missed between the top and the bottom of the graph.

- *The top-down approach.* The reasoning starts from the top and moves towards the bottom until the goal is reached. Each level is formed by inference from (as many as necessary) previous levels.
- *The bottom-up approach.* The reasoning starts from the bottom and moves toward the top until all subgoals are

satisfied. Each level is formed by the "reverse inference" from the next level (i.e., display what wold be required in order to make true the next level). (Note, this approach can result in a more effective proof than the top-down approach in many cases.)

- *The hybrid approach.* This is the combination of the top-down and the bottom-up approaches. There are two chains of reasoning; one moves from the top toward the bottom and another moves from the bottom toward the top until they meet somewhere in the middle of the graph.

4 EXAMPLES

In this section, we present several proof examples to demonstrate the top-down, bottom-up, and hybrid approaches. Example 3 deals with a proof topic in the algorithm analysis course, example 4 in the computing theory course, and example 5 in the discrete math course. Examples 6 and 7 give the proofs for the problems discussed in Section 2, and are left to the reader due to the page limit of this paper.

Example 3 (Bottom-up approach) The big-O notion and its associated proofs are an essential skill required in analysis of algorithms. Suppose we would like to prove $3n^2 + 4n - 2 = O(n^2)$; by the definition of big-O, we need to find c, $n_0 \in \mathbf{R}^+$ such that $3n^2 + 4n - 2 \le cn^2$ for all $n \ge n_0$. Figure 1 shows the step-by-step construction of the proof for this assertion. In step 1, the facts and the goal are simply depicted at top and at bottom respectively. In step 2, two subgoals are "reversely" inferred, namely, in order to make the goal true, it suffices to make each of the two subgoals true[1]. In steps 3 and 4, another layer of subgoals is reversely inferred in the same fashion. In step 5, another subgoal is added on an upper level. In step 6, the subgoal $c = 4$ can be satisfied by the fact that c is a positive number. In step 7, the subgoal $n_0 = 4$ is added which, together with the fact $n \ge n_0$, can make the subgoal below it satisfied. In step 8, finally, the subgoal n0 = 4 can be easily satisfied by the fact that n_0 is a positive number. Since both subgoals ($c = 4$ and $n_0 = 4$) on top of the reasoning chain can be satisfied, so is the original goal ($3n^2 + 4n - 2 \le cn^2$ for $n \ge n_0$), and the proof is completed.

Example 4 (Top-down approach) Formal language theory is among the most difficult materials for students to comprehend. In particular, students feel that Pumping Lemma (PL) is so abstract to grasp that using it to prove that a language is non-regular is a daunting task. Figure 2 shows the steps of the proof in our framework for the classic problem. the language $L = \{a^i b^i \mid i \ge 0\}$ is not regular.[2] In step 1, the definition of L and

[1]Note the two subgoals must be true at the same time to make the goal true. This is indicated by the arc connecting the two edges.

[2]Due to the page limit, one graph may show the combination of several steps.

the assumption that L is regular as well as the goal (to look for a contradiction) are displayed. In step 2, the (sub-)fact that there exists a $n \in \mathbf{N}$ is derived from what is assumed in step 1. In step 3, another fact is derived by the combination of two previous facts. In steps 4 and 5, two more facts are derived, the first fact is derived from a combination of two previous facts and the second fact is derived from the first fact. In steps 6 and 7, two new facts are added, both are derived as the combination two previous facts. In steps 8 and 9, the fact $xy^2z \notin L$ is derived which contradicts with the fact $xy^2z \in L$ derived in steps 6 and 7. Since a contradiction is reached, L thus cannot be regular.

Example 5 (Hybrid approach) Set theory plays a fundamental role in mathematics and computer science theory. As such, set proofs are particularly important. In Figure 3, we show how a proof for $A - (B \cup C) \subseteq (A - B) \cap (A - C)$ can be constructed. In step 1, as usual, the fact $x \in A - (B \cup C)$ and the goal $x \in (A - B) \cup (A - C)$ are displayed. In steps 2 and 3, two subgoals are reversely inferred from the goal; note that not both subgoals have to be satisfied, satisfying one would be enough. We explore the first subgoal by reversely inferring it yielding two more subgoals. In step 4, two (sub-)facts are derived from the original fact. In step 5, the subgoal $x \in A$ is satisfied by the (sub-)fact $x \in A$ obtained in step 4. In steps 6 and 7, two more (sub-)facts $x \notin B$ and $x \notin C$ are derived; note that the two (sub-)facts may not be true at the same time, but one of them must be true at any time; in the case that $x \notin B$ is true, the subgoal $x \notin B$ is then satisfied. At this point, all subgoals are satisfied and so is the goal; hence the proof is finished for the case that $x \notin B$ is true. Step 8 deals with case that $x \notin C$ is true; in this case, the subgoal $x \notin B$ is not satisfied any more which affects all subsequent inferences. Therefore, the subgoal $x \in A - C$ obtained in step 2 needs to be "turned on" and explored. In step 9, two subgoals $x \in A$ and $x \notin C$ are reversely inferred with the first subgoal being satisfied already. Step 10 shows that the second subgoal can be satisfied by the (sub-)fact $x \notin C$ obtained in step 8, and the entire proof is now completed.

Example 6 This example shows the proof for the first problem discussed in Section 2 and can be constructed in a similar way to Example 5. Due to the page limit of this paper, it is left to the reader.

Example 7 This example shows the proof for the second problem discussed in Section 2 and can be worked out in a similar way to Example 3. It is also left to the reader for the same reason.

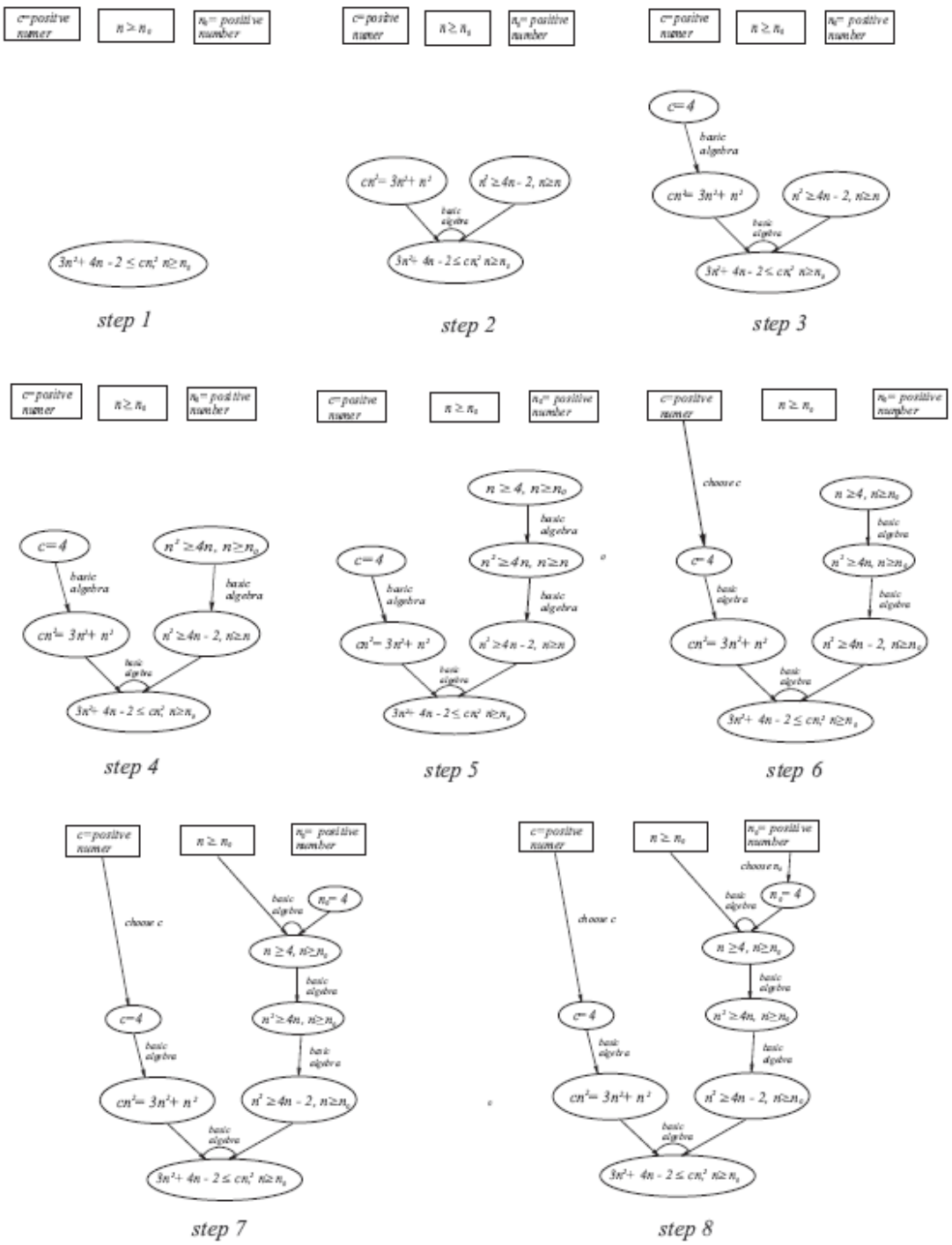


Figure 1: Proof process for $3n^2 + 4n - 2 = O(n^2)$.

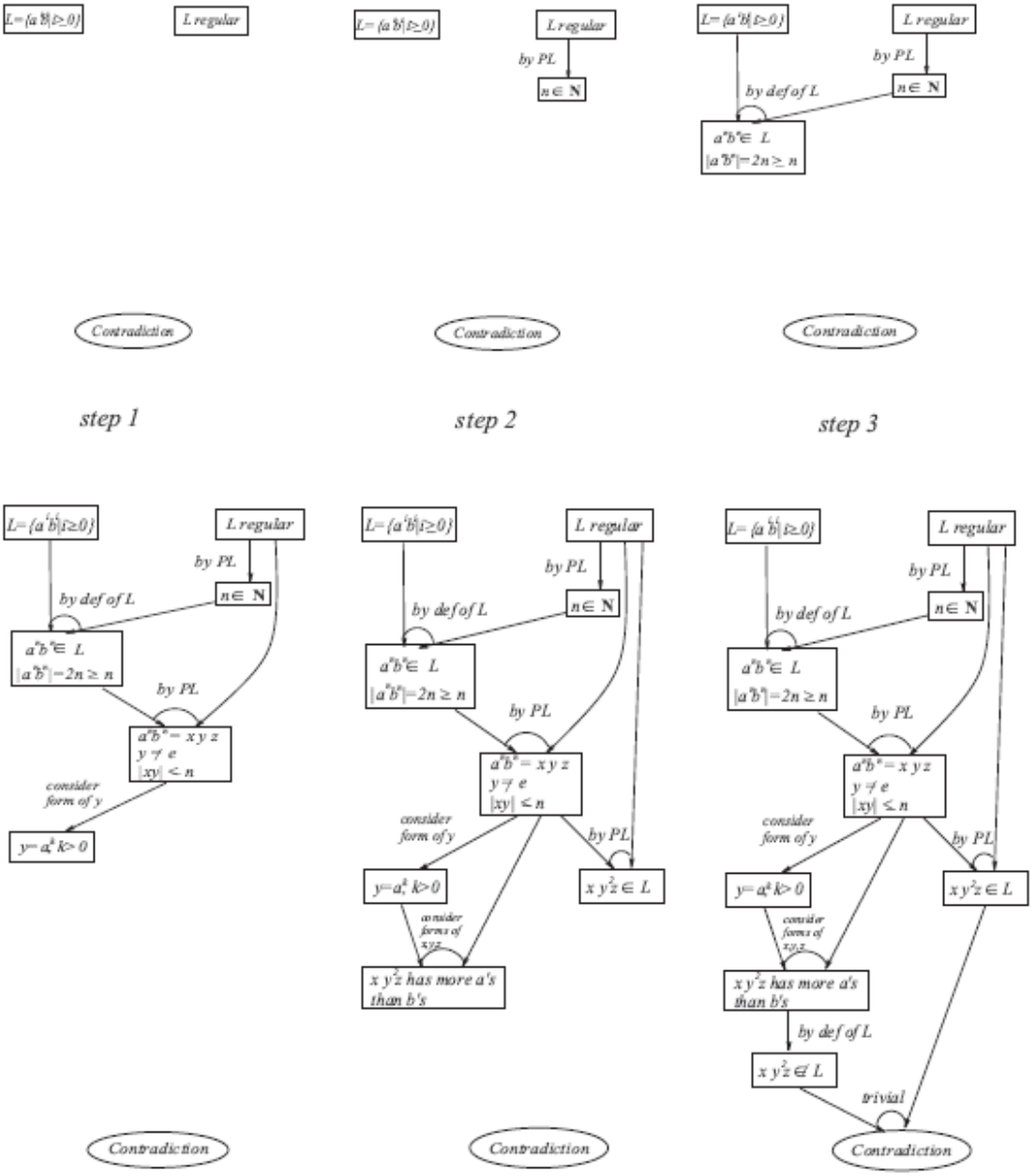


Figure 2: Proof process for showing $\{a^i b^i | i \geq 0\}$ is not regular.

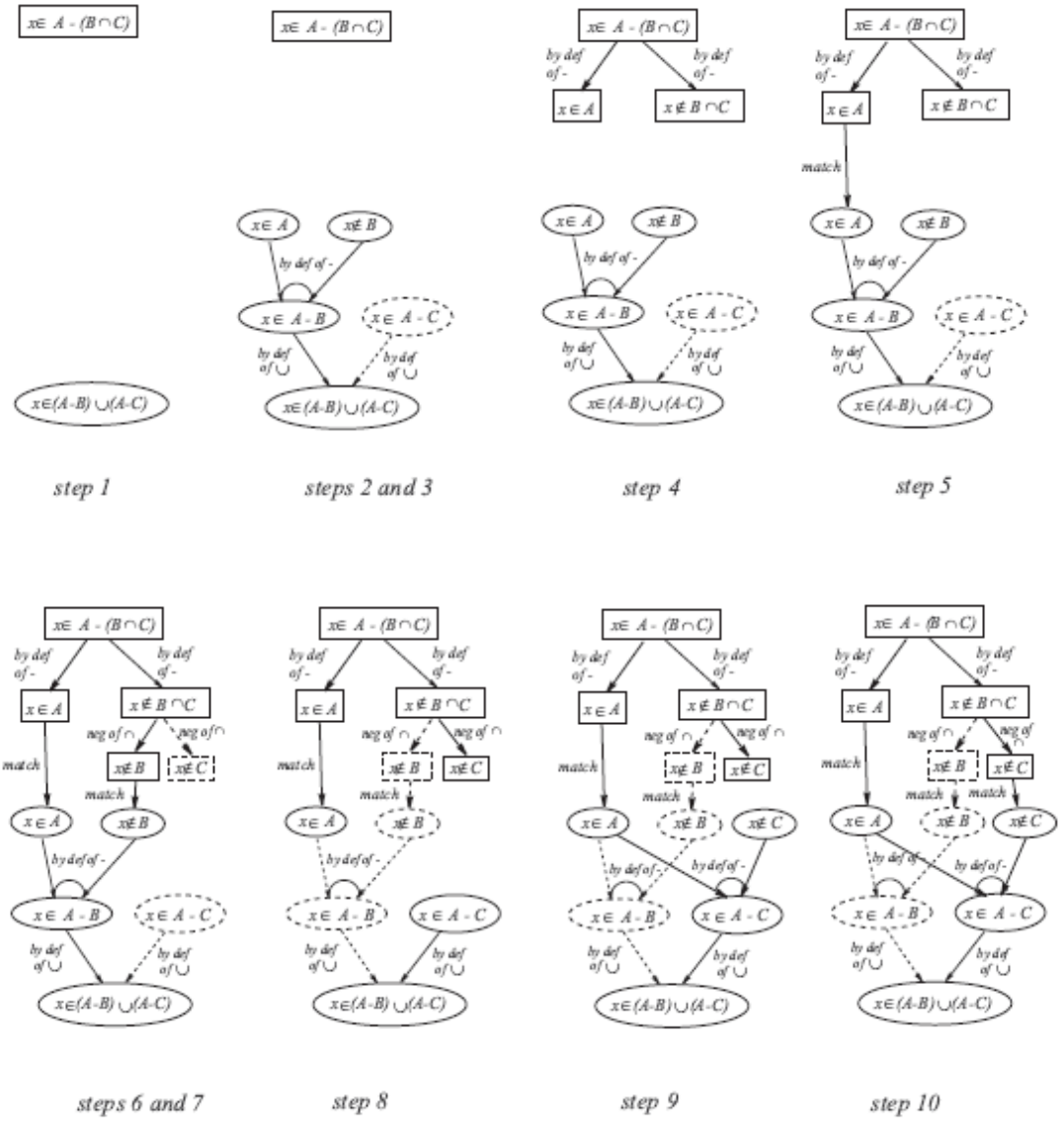


Figure 3: Proof process for showing $A - (B \cap C) \subseteq (A - B) \cup (A - C)$.

5 RELATED AND FUTURE WORK

OR graphs and AND-OR graphs are commonly used in the area of artificial intelligence (e.g. [11]) to organize the search space in a certain problem domain. Although graphical notations are used in OR and AND-OR graphs, they primarily focus on searching techniques rather than general logical inferences and proofs. It also seems that there is a somewhat obscure notion of ¡°flow proof¡± in the literature which was used to facilitate the geometry proofs in the secondary education (e.g. [10]). Compared with the flow proof, our work is more systematic, general, complete, formal, and methodological. In a sense, it is a substantial extension or rewriting of the flow proof. Indeed, our work provides a graphical framework in which proofs in any theory area can be constructed by following either the top-down, or the bottom-up, or the hybrid approach, respectively.

There are many software tools for assisting math and theory work, for example, Matlab [1], Maple [2], Mathematica [3], Coq [4], and Lillypadz [5]. However, as far as we know, none of the existing tools is able to provide any assistance for writing /constructing proofs in a graphical way as shown in this paper. Our work can serve as the initial step towards building the first math software tool that offers an interactive and user-friendly environment in which a math proof can be constructed graphically. Considering that math proof ability is a common weakness among all college students in the United States, this software tool could be significantly useful.

6 FINAL REMARKS

We have proposed a graphical framework for conducting math proofs. Compared with the traditional style of writing math proofs, we believe that this framework has the following advantages.

- *It is visual.* The logical inference structure and the relationship among various components in a (complicated) proof process are explicitly expressed as graphs.
- *It is minimum.* No extra English sentences, which could make a proof unclear if not carefully used, are needed to help describe the proof. A proof under this framework is a big logical formula written in a graphical manner.
- *It is a template.* It provides students an effective and easy-to-follow guidance in terms of learning how to write/construct proofs.

REFERENCES

[1] http://www.mathworks.com/.

[2] http://www.maplesoft.com/Products/Maple/.

[3] http://www.wolfram.com/.

[4] http://coq.inria.fr/.

[5] http://lillypadz.com/.

[6] Code breakthrough delivers safer computing. http://www.unsw.edu.au/news/pad/articles/2009/sep/, 2009.

[7] K. Devlin. Why universities require computer science students to take math (spe. session). *CACM*, 46(9), 2003.

[8] Jeff Kramer. Is abstraction the key to computing? *CACM*, 50(4), 2007.

[9] H. Lewis and C. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 2nd edition, 1998.

[10] Robert McMurray. *Geometry: A Flow Proof Approach*. 1976.

[11] Elaine Rich and Kevin Knight. *Artificial Intelligence*. McGraw-Hill, 2nd edition, 1991

UNDERSTANDING NSF FUNDING OPPORTUNITIES*

TUTORIAL PRESENTATION

Scott Grissom
National Science Foundation
4201 Wilson Blvd, Suite 835
Arlington, VA 22230
(703) 292-4643
sgrissom@nsf.gov

ABSTRACT

This session highlights programs in the National Science Foundation (NSF) of particular interest to computer science educators. Topics include a description of program goals, guidelines, review process as well as strategies for writing competitive proposals.

INTRODUCTION

NSF supports projects to improve education in science, technology, engineering, and mathematics through several programs in its Education and Human Resources (EHR) directorate, as well as in its research directorates, including Computer and Information Science and Engineering (CISE). This tutorial presents a description of some education-related programs in the EHR and CISE directorates, and enables participants to interact with the presenters concerning specific project ideas that could be appropriate for the various programs.

SPECIFIC PROGRAMS DISCUSSED

Complete details about each of the following programs can be found on the NSF websites for the Division of Undergraduate Education (DUE) [1] and the Directorate for Computer & Information Science & Engineering (CISE) [2].

- Course, Curriculum, and Laboratory Improvement (CCLI)
- CISE Pathways to Revitalized Undergraduate Computing Education (CPATH)
- Federal Cyber Service: Scholarships for Service (SFS)
- Research Experiences for Undergraduates Sites (REU Sites)

- Broadening Participation in Computing (BPC)
- Scholarships in Science, Technology, Engineering and Mathematics (S-STEM)
- STEM Talent Expansion Program (STEP)
- Advanced Technological Education (ATE)

WRITING COMPETITIVE PROPOSALS

NSF programs are quite competitive but there are simple strategies that investigators should be aware of to improve their chances of success. First and foremost, read the Program Solicitation carefully. The goal is to help reviewers quickly understand *what* you intend to do and that you have given sufficient thought of *how* you intend to do it. Organize the proposal to address the essential components described in the solicitation. Use headings, boldface and bulleted lists to help the reader quickly understand the organization of the proposals. Address each point thoroughly but succinctly. And finally, start well before the submission deadline. Include sufficient lead time to allow colleagues to provide feedback and for your research office to approve the proposal. Specific directions for completing a proposal can be found online [3].

REFERENCES

[1] NSF Division of Undergraduate Education (DUE), http://nsf.gov/div/index.jsp?div=DUE, 2009.

[2] NSF Directorate for Computer & Information Science & Engineering (CISE), http://nsf.gov/dir/index.jsp?org=CISE, 2009.

[3] NSF Funding, How to Prepare Your Proposal, http://nsf.gov/funding/preparing, 2009.

GAPS IN THE COMPUTER SCIENCE CURRICULUM: AN EXPLORATORY STUDY OF INDUSTRY PROFESSIONALS *

Chris B. Simmons
Department of Computer Science
University of Memphis
Memphis, TN 38152
901-678-5465
cbsmmons@memphis.edu

Lakisha L. Simmons
School of Business Administration
University of Mississippi
University, MS 38677
662-915-5777
lsimmons@olemiss.edu

ABSTRACT

Our study explores current industry needs in suggesting how to better prepare computer science graduates with the appropriate background that will enable a successful career. With the increase in outsource development, computer scientists find themselves in some project management related capacity. Unfortunately, computer science graduates lack the interpersonal skills needed to successfully fulfill duties associated with outsourcing. We report findings from qualitative interviews from IT professionals in Fortune 500 businesses, small-to-medium businesses, and non-profit organizations. Our analysis concludes that modifying computer science curriculum to provide more emphasis on negotiation skills, time management, cultural differences, outsource management, and information assurance would make the most difference, in addition to a strong technical background.

INTRODUCTION

The computer science curriculum has continued to evolve throughout the years within universities and colleges, even more so now with the high rate of outsourcing and drop in computer science enrollments [5]. It is essential to provide a curriculum that improves the industry as well as creates a well-rounded computer scientist. Patterson [7] highlighted the need for computer science (CS educators to provide a curriculum that

reflect opportunities and challenges of information technology in contemporary times. Therefore, is the CS curriculum relevant for today's industry needs? To date, there has been a lack of research focusing on the current workforce needs for computer science graduates.

Recent news indicates that industries that hire computer science graduates are changing in this time of hiring freezes, low budgets, and increased outsourcing for decreased costs. Some age old issues also still exist [e.g. 3]. In 2007, Havill and Ludwig [4] stated computer science undergraduates have difficulty expressing their work to a general audience due to the highly technical nature. This provides valuable insight into the need to add focus on communicating in nontechnical terms more effectively.

Universities may teach students more about the development of technologies [8] as well as soft skills such as determining what to develop and how to communicate during development. There is a lack of research involving actual curriculum enhancements based on industry needs. For example, how to teach computer scientists to ask the right questions when attempting to contribute on a project team. Fitzpatrick [2] stated bridging the gap involves students experiencing projects in action to help provide a needed understanding of software development. We suggest curriculum enhancements based on such literature and exploratory interviews with 20 IT professionals in a number of industries. We asked how CS graduates are used in their companies and any suggestions for curricula enhancements so CS graduates could be more productive and useful.

CURRENT CURRICULUM

Research into the computer science curriculum has suggested various phases of software engineering are needed to improve students' knowledgebase when entering the workforce. Some literature suggests introducing extreme programming, testing methodology, and communication skills. Patterson [7] addressed the need to include open source software into the curriculum, as it provides graduates with the ability to help improve potential organizations. The need to present computer scientists with open source tools, such as Eclipse, enables CS graduates with near real world tools, contrary to programming on the command line. Douglas et al. [1] provided insight to the need of incorporating human computer interaction into the computer science curriculum, as 50% of code written for software applications is designed for the user interface. With this knowledge, it is becoming increasingly important for computer scientists to effectively elicit customer requirements to prevent rework. Fitzpatrick [2] introduced the Software Quality Star, a model for improving software quality motivated by ISO/IEC 12207, which attempts to integrate human-computer interaction (HCI) and software engineering to improve curriculum regarding customer usability. Havill and Ludwig [4] suggested three approaches to improving CS undergraduate programs, which include improving communication skills, exposing students to research early, and increasing exposure to mathematics. This is a valid point for computer scientists moving towards a graduate level degree, but lacks the increasing need of how to effectively work with customers and understand what is being requested. We seek to understand the current gaps in CS curriculum and what the industry needs from CS graduates.

METHODOLOGY

In order to understand what industry requires from newly minted computer science graduates, we conducted 20 interviews that consisted of 7 open-ended questions related to what technical industry professionals described as being an important need for future computer science professionals. Our interviews were conducted with IT professionals such as 6 Sigma Black Belts, Senior Project Analysts, Quality Assurance Analysts, Computer Scientists and one president of a nonprofit organization. The professionals were from 8 Fortune 500 businesses, 9 small-to-medium businesses, and 2 non-profit organizations. So far, we completed 20 interviews that provided insight to where organizations see the future of computer science and potential areas of focus in the computer science curriculum. We asked questions based on the literature above discussing outsourcing, soft and behavioral skills, and duties of CS graduates. In the results section we provide a summary of each question from each professional.

RESULTS

In this section we provide a summary of the interviews conducted with the IT professionals.

1. *What positions does your company usually fill with computer science graduates (not MIS or business information systems majors)?*

   There is a consensus from the interviews that a wide range of positions are available for computer scientists. Positions consisted of programmer analyst, software engineer, test engineer, network admin, telecommunication, application development, quality engineer, systems engineer, requirements engineer, project engineer, operation support, help desk, technical advisor, software architect, technical analyst, and business analyst.

2. *How important is it for computer science undergraduates to know how to gather and elicit customer requirements to fulfill IT positions in your company?*

   From the survey we find that gathering and eliciting customer requirements is very desirable skill set for computer science graduates. 80% of professionals surveyed stated the importance was high, where 20% stated the importance was medium to low. Those in the minority state a senior level person performs this job, but this further solidifies the need for computer science curriculum to include requirement elicitation techniques to create a well-rounded graduate when entering corporate America.

3, *What do computer science undergraduates need to have in their undergraduate programs in terms of soft/behavioral skill sets?*

Participants suggested computer science curricula need to include training on effective writing skills for documentation and status reporting and communication and presentation skills. Team building, the ability to be flexible and how to deal with hostile personalities were highly emphasized. Negotiation skills, software requirement gathering, project management, and information management skills were also mentioned as important soft skill sets.

4. *What do computer science undergraduates need to have in their undergraduate program in terms of technical skill sets (e.g. programming languages)?*

   Our interviewees discussed the importance of object oriented programming, decoupling understanding, analytical skills, and integrity in programming. Specific languages such as Java, JavaScript, Ruby, Perl, HTML, CSS, SQL, Python, JUnit, .Net, C++, and UML were identified. Other concepts such as web services, network administration, server administration, database administration, and requirement engineering were stated.

5. *What do computer science undergraduates need to have in their undergraduate program to help them assist with your company's outsourced development efforts?*

   Results suggest that more universities allow companies to come in to explain their needs. Others suggest the following training opportunities to assist in outsourced development: database knowledge, shell scripting, swing, configuration management, process improvement, unit testing, documentation, project management, time management, cultural differences, outsource management, information assurance, and tools and technologies that will help improve software during the software development lifecycle.

6. *What do computer science undergraduates need to have in their undergraduate program to help them when they are required to work on project teams (e.g. process improvement teams or new product development teams)?*

   Interview participants recommend the need for CS students to be familiar with process improvement frameworks (e.g. 6 Sigma and CMMI). Significant group project related skills, good communication, time forecasting, and how to ask the right questions were some of the most prominent comments for this questions. The systems development life cycle was described as a fundamental piece of CS curriculum but Microsoft Project, RUP, and PMBOK, placed in the curriculum could enable success when entering the workforce in any position.

7. *In general, what do computer science undergraduates need to know, to enable them to have a successful career in an IT department?*

   Participants mentioned that CS graduates need to know how their work affects the bottom line. Doing so could help them realize that their work matters and their best

effort should always be put forth. Some characteristics mentioned include being flexible, being able to meet short and long term goals, communicating with upper management, having integrity and a great attitude. A wide knowledge of systems and technology integration, certificates on the latest technology, open source technologies, as well as being a good researcher, and obtaining a good mentor would lead to success in today's IT department.

CONCLUSION

This paper has presented preliminary, yet valuable information to improve the computer science curriculum in preparing graduates for today and tomorrow's positions. This exploratory study has been conducted using rich interviews from IT professionals in various types of organizations. Our analysis concludes that modifying computer science curriculum to provide more emphasis on written and verbal communication skills, gathering and eliciting customer requirements effectively (80% of respondents emphasized), the ability to be flexible and the ability to deal with varying personalities were highly emphasized. Negotiation skills, time management, cultural differences, outsource management, and information assurance trainings were some of the most notable skills in addition to a strong technical background. All of these skills will aid in improving graduates' ability to communicate on all levels of the organization when entering industry in an IT capacity position. This is pertinent as companies continue to outsource. The ability to provide well-rounded computer scientists is vital in these situations. With the current literature aiming to improve various aspects of the computer science curriculum, we feel there is a grave need to focus on the customer. We propose enhancing the computer science curriculum to include more customer elicitation techniques, albeit potentially through a business, to improve the ability for computer scientists to hone in on the art of requirement elicitation and project management through communication skills.

Future research can further quantify the needs of industry. Then focus on how to successfully implement these important skill sets into CS curriculums. Requirements elicitation, communication, and project management skills within the computer science curriculum are some of the potential areas to make a great impact in the career success of computer science graduates.

REFERENCES

[1] Douglas, S., Tremaine, M., Leventhal, L. and Wills, C.E. Incorporating human-computer interaction into the undergraduate computer science curriculum, In *Proceedings of the Technical Symposium on Computer Science Education (SIGCSE)*, 211-212, 2001.

[2] Fitzpatrick, R. The Software Quality Star: A conceptual model for the software quality curriculum. In Closing the Gaps: Software Engineering and Human-Computer Interaction, Harning, M. and Vanderdonckt, J., 9-13, 2003.

[3] Giangrande, E., Communication skills in the CS curriculum, *Journal of Computing Sciences in Colleges,* 24, (4), pg. 74-79, 2009.

[4] Havill, J. T. and Ludwig, L. D. Technically speaking: fostering the communication skills of computer science and mathematics students, In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* March, 2007.

[5] Kahlon, A. and Lee, L. Essential programming concepts: the redesign of the introductory programming course. *Journal of Computing Sciences in Colleges,* 24, (4), 53-53, 2009.

[6] McCrickard, D. S., Chewar, C. M., and Somervell, J. Design, Science, and Engineering Topics?: Teaching HCI with a Unified Method. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education (SIGCSE)* ACM Press, New York, NY, 31-35, 2004.

[7] Patterson, David A. Computer science education in the 21st century, *Communications of the ACM 49, (3), 2006.*

[8] Walker, Henry M., Schneider, M. A revised model curriculum for a liberal arts degree in computer science, *Communications of the ACM*, 39, (2), 1996.

WHY COMPUTATIONAL THINKING SHOULD BE INTEGRATED INTO THE CURRICULUM*

Jake A. Qualls and Linda B. Sherrell
University of Memphis
Department of Computer Science
Memphis, TN 38152
(901) 678-5465
{jaqualls@; Linda.Sherrell@}memphis.edu

ABSTRACT

Computational Thinking (CT) is an approach to problem solving that consolidates logic skills with core computer science concepts. This survey paper reviews recent efforts to integrate CT into primary, secondary and post-secondary curricula. The paper should prove beneficial to instructors interested in investigating this important topic.

1. INTRODUCTION

Computer science is the theoretical evaluation of computation. Algorithmic processes are analyzed and designed to computationally model and solve problems. The need for problem solving techniques is inherent in all scientific and engineering disciplines [11]. Improvements in computing have allowed these fields to incorporate computational techniques; however, teaching professionals that want to take advantage of computational advancements will need to adjust their current perception of thinking skills.

Jeannette Wing's seminal paper "Computational Thinking" (CT), proposes a way to incorporate computing into all academic fields by emphasizing "a range of mental tools that reflect the breadth of the field of computer science" [11]. Wing defines CT as the use of computer science concepts to solve a problem in any domain. To help clarify the notion of computational thinking, Wing lists six principles: 1) CT is conceptualizing via abstraction; 2) CT is a fundamental skill needed to function in modern society; 3) CT

does not necessarily mean that problems must be solved like a computer, rather it encompasses all critical skills of humans; 4) CT complements and combines mathematics and engineering; 5) CT is principally concerned with ideas as opposed to artifacts; 6) CT should be an integral part of everyone's education. In summary, Wing's article provides a grand vision for computational thinking.

In [4], Guzdial stresses that educators need to make CT accessible to everyone. Specifically, curriculum changes may need to occur to expedite the process. He identifies questions that need clarification in order to make advancements within the field, such as:

- What do non-computing students understand about computing?
- What will they find challenging?
- What kinds of tools can make computational thinking most easily accessible to them?
- How should we organize and structure our courses to make computing accessible to the broad range of students?

He also emphasizes educators must understand better how to teach computing so that students have the knowledge to successfully apply computing in their fields of interest.

In 2009, the Computer & Information Science & Engineering (CISE) division of the National Science Foundation (NSF) recognized the importance of CT and how it plays a critical role in education and society. In particular, CT is now a required component of all CISE Pathways to Revitalized Undergraduate Computing Education (CPATH) grant proposals, as documented in the current solicitation. It is important to note that this modification to the program acknowledges the importance of CT not only in computer science but in all disciplines.

2. INTRODUCTORY COMPUTER SCIENCE SEQUENCE

2.1 CT in Early CS Courses

The Tri-P-LETS (Three P Learning Environment for Teachers and Students) outreach project at the University of Memphis focused on problem solving skills, programming concepts and a software development process [9]. The project, which was funded by NSF from 2004 to 2008, involved 11 high schools from two school districts. Students could enroll in two years of coursework. First year students worked in teams to plan, program and document projects using AgentSheets, a game authoring and simulation-building tool. Object-oriented concepts were taught in a gradient fashion using learning modules that incorporated concepts from a variety of disciplines. For instance, one module required students to model the progression of the water cycle using AgentSheets, while another simulated the spread of viruses. Students were also introduced to more advanced computer science topics such as cellular automata and combinatorics. The second course used Java and Lego robots to emphasize object-oriented concepts. In summary, students gained hands-on experience by building simulations and programming robots, while learning important computing skills and reviewing concepts from other scientific domains in the process.

In 2006, Carnegie Mellon University initiated an outreach program called Computer Science for High Schools (CS4HS). This program provides high school computer science

teachers with material that emphasizes computational thinking [1]. Program goals include an increased awareness among teachers and provisions for integrating computer science into different fields. At the first workshop, a survey revealed the majority of high school teachers believed computer science to be synonymous with programming. These findings are similar to those collected from a survey of 800 high school students [2]. Participating teachers named several reasons for the decline in high school computer science enrollments: outsourcing reports, the dot-com bubble, ill-prepared teachers and lack of understanding by administration. In a post-workshop questionnaire, teachers demonstrated a better understanding of computer science. Note that CS4HS has recently been expanded to aide faculty from colleges and universities to initiate similar programs.

In [7], Lu and Fletcher argue that CT proficiency should begin in primary and secondary curricula with the teaching of vocabularies and symbols to form a solid foundation for further skill development. They equate programming with the proof construction found in mathematics. Unfortunately, theorem proving, which is a key skill for computer scientists, is usually not mastered by students until later in the CS curriculum. To remedy this problem, Lu and Fletcher propose a computational thinking language (CTL). CTL introduces concepts such as state, data, iteration, searching and efficiency. To handle more complex problems, the language includes basic tuple notation, which familiarizes students with the concepts of elements and ordered lists. By adopting Lu and Fletcher's approach to teaching programming concepts, teachers can better prepare students who choose computer science as a field of interest.

2.2 CT in College Courses

Integrating computational thinking within a CS curriculum requires the reformulation of teaching techniques used within freshman level courses. One method of change is to incorporate real-world analogies that aide in the understanding of core computing concepts. Sooriamurthi [10] explores the object-oriented paradigm from the perspective of an introductory CS0 course by providing "a conceptual 10,000 foot view of the essential ideas of object orientation that is independent of any particular OO language." To establish and illustrate object-oriented concepts, both scientific and pop-culture references are employed. As an example, the anthropomorphic nature of Disney's Bedknobs and Broomsticks eases students into the ideas and concepts of object oriented thought. By incorporating anecdotes to which students can easily relate, the concepts become memorable and form a solid foundation of core programming components.

Alice is a popular learning environment for introducing and teaching programming concepts in CS0. In [3], Dougherty describes six virtual worlds created with *Alice* to engage students while stressing important algorithmic concepts. His approach relies on the visual nature of *Alice* to facilitate the knowledge acquisition of students. The three-week module consists of lectures using virtual worlds to illustrate concepts along with three mandatory lab sessions. The first virtual world introduces methods, functions, parameters, iteration, recursion and generalizing a solution. The second week uses a new virtual world to expand on looping, to introduce arrays, and to emphasize the distinction between methods and functions. To complete the three-week module, the last virtual world focuses on lists, sorting and event interaction. During each lab session, students complete a virtual world to reinforce concepts discussed during lecture. Performance was

found to be comparable to the previous version of the course that used JavaScript. Furthermore, the students are "engaged to a greater degree with *Alice*, motivated to design and implement substantial term projects and (sometimes) continue to CS1."

Haden and Mann discuss difficulties encountered when teaching programming and define a course sequence to "act as a bridge between traditional procedural programming and advanced Object-Oriented system development" [5]. The course sequence was designed to emphasize basic programming skills, because the instructors had previously found prolonged procedural experience made the shift to an Object-Oriented paradigm difficult for their students. The sequence includes complex, but engaging problems that seamlessly build on previous skills. The first course emphasizes syntax, variables, and program logic using Pascal, while the second course introduces object-oriented concepts through Object Pascal. The development environment, Borland Delphi, allows students to efficiently construct graphical user interfaces. With this approach, students are able to quickly produce games, while learning concepts such as the event-driven interface model. Haden and Mann identified syntax errors and method implementation as the most common types of errors in their approach.

3. INTERDISCLIPINARY APPROACHES

The interdisciplinary acceptance of Computational Thinking can be readily observed within the field of bioinformatics. Qin elaborates on the experiences of developing an introductory bioinformatics course at Tuskegee University [8]. He notes that for scientists to become productive in the 21st century, they need to embrace the concepts of computing. Qin relates that teaching CT to life science students presented a paradoxical situation as "most biology students readily express their interests in improving their computational skills, however, few of them actually take computing-oriented courses." To organize his course, a direct mapping between key biology concepts and CT topics was made. This exercise provided a blueprint for creating ideal projects suited for the course. Faculty found non-computing students were intimidated by interacting with computers and remedied the problem by creating peer-programming groups. To maintain accountability for individual work, the instructor deployed the following: 1) students were forced to change groups often; 2) they were required to submit individual reports when working; 3) quizzes and in-class discussions were held to emphasize understanding. Post-course surveys revealed students were satisfied with the course organization. They also expressed positive feelings about the learning experience. Qin notes some students went on to enroll in additional computing courses.

The initial assessment of a newly developed course in CT for science majors appears in [6]. Hambrusch, et. al. argue that computing has become an integral part of research within the scientific community, because humans can no longer process the amount of generated data in an efficient manner. In collaboration with various science departments, a course was developed that utilizes a problem-based approach while emphasizing scientific discovery with computer science principles. The main objective of the course is to create a firm understanding of computational ideas that students can apply to their own scientific investigations. In addition, the course presents examples and projects catered to students by using language familiar to their scientific disciplines. Post course

surveys indicated that 60% of the enrolled students intended to take additional CS courses, while 40% planned on pursuing a minor in CS.

4. CONCLUSIONS AND FUTURE WORK

This paper has illustrated some recent work conducted in an effort to incorporate computational thinking into curricula. Tri-P-LETS and the CS4HS outreach programs are prime examples. However, the idea of CT is to integrate computational techniques and approaches into all disciplines requiring problem-solving skills. To thoroughly transform CT into one of the four primary skills, (reading, writing, arithmetic, and computational thinking), CT activities must appear as early as the primary grades, and then continue through the secondary grades and beyond. Recent recognition by the NSF proves that CT is an important component for education and society and deserves our immediate attention. Therefore, this paper has also discussed interdisciplinary approaches.

Resolving the underlying misconception that equates CS with *programming* is the first step in paving the way for CT. Empowering teachers with valid information about CT provides an opportunity that could inspire high school students to explore computing with renewed interest. Results from computing education research enable educators to measure the learning levels to make necessary curriculum changes to broaden accessibility to all students.

This paper will be of benefit to instructors who wish to conduct their own literature review of Computational Thinking, as it provides an initial survey of recently published work in the field. The authors note that publications exist that do not explicitly mention CT but have similar aims to those expressed by Wing. Future work will describe the integration of CT into the introductory computer science sequence at the University of Memphis.

5. REFERENCES

[1] Blum, L., Cortina, T. J., CS4HS: an outreach program for high school CS teachers, *38th SIGCSE Technical Symposium on Computer Science Education*, 19-23, 2007.

[2] Carter, L., Why students with an aptitude for computer science don't choose to major in computer science, *37th SIGCSE Technical Symposium on Computer Science Education*, 27-31, 2006.

[3] Dougherty, J. P., Concept visualization in CS0 using ALICE, *Journal of Computing Sciences in Colleges, 22*, (3), 145-152, 2007.

[4] Guzdial, M., Paving the way for computational thinking, *Communications of the ACM*, *51*, (8), 25-27, 2008.

[5] Haden, P., Mann, S., The trouble with teaching programming, *16th Annual National Advisory Committee on Computing Qualifications (NACCQ),* 63-70, 2003.

[6] Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., Hosking, A. L., A multidisciplinary approach towards computational thinking for science majors, *40th SIGCSE Technical Symposium on Computer Science Education,* 183-187, 2009.

[7] Lu, J., Fletcher, G., Thinking about computational thinking, *40th SIGCSE Technical Symposium on Computer Science Education,* 260-264, 2009.

[8] Qin, H., Teaching computational thinking through bioinformatics to biology students, *Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education*, *41*, (1), 188-191, 2009.

[9] Sherrell, L. , Liu, C., Pottenger, W., Gross, P., NSF-DGE GK-12 teaching fellowships: changing student perceptions about computer science, *SIGCSE 2007: 38th Technical Symposium on Computer Science Education*, *39*, (1), 474-475, 2007.

[10] Sooriamurthi, R., The essence of object orientation for CS0: concepts without code, *Consortium for Computing Sciences in Colleges*, *25*, (3), 67-74, 2010.

[11] Wing, J. M., Computational Thinking, *Communications of the ACM*, *49*, (3), 33-35, 2006.

HIVE: CROWDSOURCING EDUCATION DATA*

Neal Gibson and John Talburt
Department of Information Science
University of Arkansas at Little Rock
Little Rock, AR 72204
501 371 7616
neal.gibson@arkansas.gov jrtalburt@ualr.edu

ABSTRACT

Data Driven Decision Making is a well-worn mantra in education, best personified by the enormous amount of data created by the yearly student assessments of literacy, math, and science mandated by No Child Left Behind. The U.S. Department of Education has recently begun giving states grants to create longitudinal data systems. The purpose of these grants is to help states organize and present these data in meaningful ways so that they can be used to improve student achievement. However, little research has been done on best practices on how educators should use these data. The Arkansas Department of Education initiated a year-long study on data use by educators and, based on this research, has developed an interactive data visualization application that incorporates social networking tools which allow educators to collaborate on data analysis.

INTRODUCTION

The National Center for Education Statistics' Institute of Education Sciences, in its Statewide Longitudinal Systems Grant Program, will have awarded $500,000,000 to states, territories, and the District of Columbia once its third round of funding is finalized early in 2010. These grants are intended to help "states, districts, schools, and teachers make data-driven decisions to improve student learning."[1] In its "Forum Guide to Decision Support Systems," the NCES outlines a general approach to how such a data system should be constructed, but only a single page is devoted to user training, and this discussion is entirely devoted to planning for training and not specific in any way to best practices around data use.[2]

While No Child Left Behind (NCLB) and the accountability it represents for schools makes Data Driven Decision Making (DDDM) a priority and generous federal grants mean it is also well funded, there are many unanswered questions about how educators should interpret and analyze data and about the effects DDDM is having on educational outcomes.[3] There are some general guides for implementing DDDM, most modeled after Edward Deming's Plan, Do, Study, Act cycle. For example, Harvard's "Data Wise" improvement process lists eight steps, under the general categories of Prepare, Inquire, and Act.[4] Victoria Bernhardt's model, Education for the Future, is a bit closer to Deming's original model with seven steps under the categories of Plan, Implement, Evaluate, and Improve, but these steps are organized around another central category of Vision.[5] While both of these models represent the current thinking on how best to use data in education, neither offer any specifics on how to implement these programs or any metrics to judge the efficacy of the approach in improving student outcomes.

More comprehensive models of DDDM have been proposed. Mandinach, Honey, and Light propose a framework whereby individuals at different levels of the hierarchy have questions or problems which require data to be collected and analyzed in order to make informed decisions.[6] Ikemoto and Marsh expand on this framework to differentiate both "simple and complex data" and "simple and complex analysis and decision making."[7] Ikemoto and Marsh are somewhat dismissive of the Mandinach, Honey, and Light framework, suggesting that it is not applicable to the nuances and variation of decision making in real-world settings and that decision making is not always as linear or continuous as they propose. What Ikemoto and Marsh fail to recognize is that even "simple analysis and decision making" is fraught with potential problems. Even simple analysis can lead the data consumer down erroneous paths if the common errors of data analysis are not first recognized and somehow accommodated for in how the data is presented.

What is somewhat assumed in all the aforementioned methodologies for analyzing data is that educators somehow have an innate ability to analyze data, so more data is always better. As Simon points out, we do know that information consumes attention, so this "wealth of information creates a poverty of attention."[8] What is missing in these discussions is an abundance of research that documents the elementary errors even intelligent consumers can make when analyzing data and making decisions based on these data. Stanovich has labeled such errors "fundamental computational biases" because they are innate and pervasive.[9] Any attempts at helping educators make decisions from data must first begin with an analysis of how they interact with data and the types of mistakes they make when analyzing data.

Background

The Arkansas Department of Education (ADE), in partnership with the Assessment and Accountably Comprehensive Center, conducted a year-long pilot project to study how educators from five different schools use data to make decisions. ADE has received numerous accolades for its data system, for example, receiving an "A" for educational data from the U.S. Chamber of Commerce.[10] However, this research was the first to actually analyze how educators interact with data. Multiple visits were made during the year, and each session introduced data with increasing granularity, starting with state

yearly assessments and eventually ending with the educators analyzing local, more frequent data.

The tendency for educators to contextualize and socialize their analysis was seen as a consistent problem. For example, a single point of data from a single year would be seen as confirming that a particular teacher, textbook, or program was not effective, depending on the user's own perspective. It was also discovered that the data system did not provide data to users in a way that facilitated multiple forms of analysis. It was very difficult for users to go from aggregate details to student-level data. It was impossible for educators to combine their own local data with data available from the state's system. They did not have their own database to do this, and no one at their schools had the technical ability to create one for them. Educators could also not objectively identify student growth, other than to simply look at the movement of a student from one NCLB category to the next. The lack of time available for teachers to come together for data analysis was also identified as a major hurdle.

Project Goals

The goals of this project were to address each of the problems identified in the pilot project and incorporate participant suggestions on how to modify the system. It was decided that a new visualization tool should be created that would allow educators to more easily manipulate data. The tool must also allow educators to upload their own local data and use the same visualization tools for analysis of local data. To help alleviate problems associated with educators having little time to come together for data analysis, it was decided that social networking tools be included in order to facilitate asynchronous collaboration. To prevent contextual biases from corrupting the analysis, a stepwise framework and protocol for data analysis, specific to the type of data being examined, would need to be developed.

METHODOLOGY

The first step was to create the data analysis framework and protocol and then build the visualization tool to support the framework. The beginning reference for this framework was from the much more general work of Van Houten et al, but the framework was modified to be specific for Arkansas educators.[11] Each section begins with a specific question to be answered, along with a methodology for determining the answers. Educators use the framework to step through the data analysis process, beginning with "big data," represented by the state annual assessments, and ending with "little data," the much more frequent and timely local data. A protocol for writing data statements is included to make sure that educators stay objective in their analysis.

The visualization tool was built using the prefuse visualization toolkit.[12] A brushing class was added to allow users to quickly visualize various NCLB categories. A zoom feature was added to allow users to "drill down" and "drill up." New visualization were added, including a bubble chart, scatter bar, bar chart with box plots, and an animated scatter plot which shows a cohort over time. Basic social networking functionality with threaded discussion was also built. This was added to facilitate the

creation of online, asynchronous professional learning communities around the analysis of data.

The system is called "hive," in the sense of "hive computing," but in this case, it is human computation. We are "crowdsourcing" the analysis of education data--harnessing the power of collective intelligence for the task of finding patterns in the data that would be missed by traditional data mining techniques. In crowdsourcing, every individual possesses some specific knowledge that can serve the needs of others,[13] In the case of education data, aggregate values may give educators and policy makers some indication concerning trends, but those that are much closer to the problem, such as teachers and administrators that know these data as individual students and programs of support for these students, can provide much richer detail through the context they can provide. If there are significant trends suggested by the data, it would be the local educators that can best identify possible program or personnel changes that might explain the differences.

The website is available at http://hive.arkansas.gov and provides two levels of access, public and authorized educators. Here are some example visualizations:

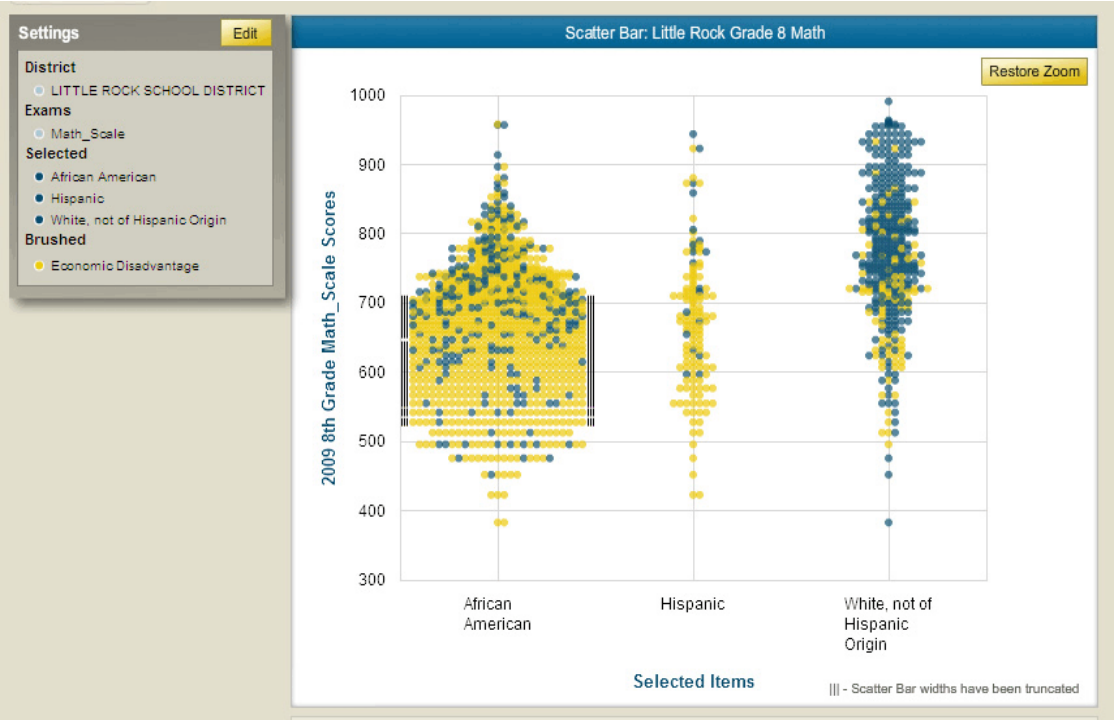


Figure 1. Scatter Bar of Grade 8 2009 math scores in Little Rock by ethnicity. The number of White students performing at the higher levels is clearly obvious, but the category "Economic Disadvantaged" is brushed, which suggests the performance difference is related more to poverty than to ethnicity.

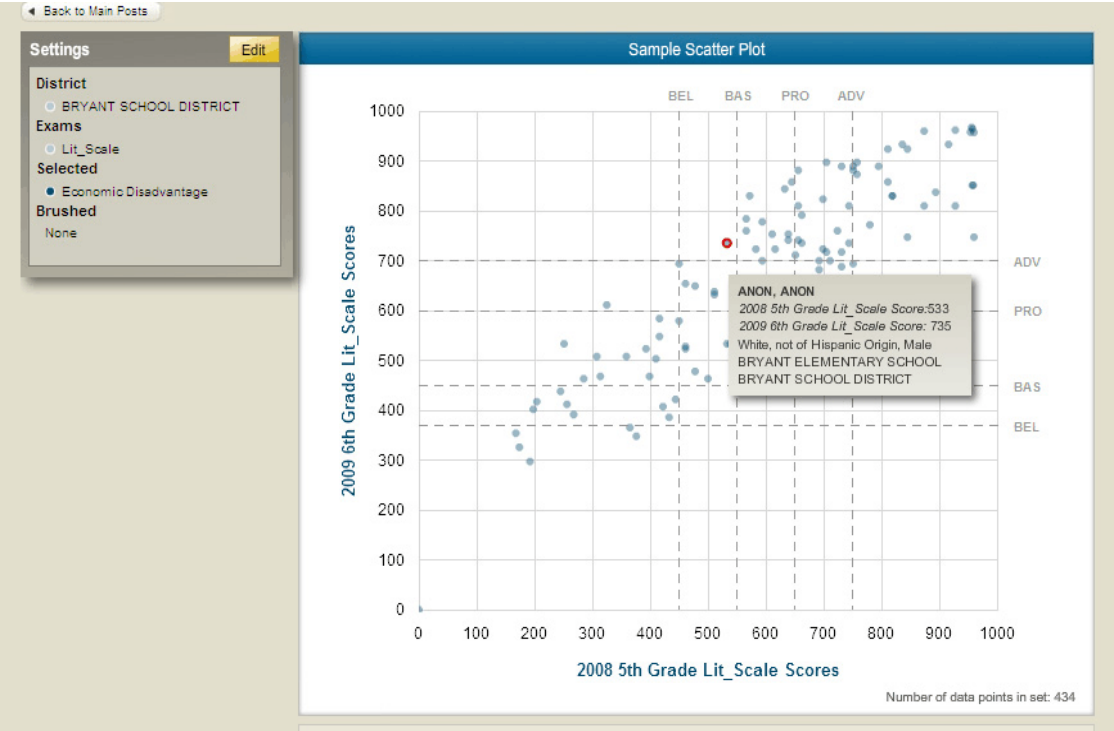


Figure 3. Scatter Plot of Grade Five 2008 and Grade Six 2009 literacy scores for students in Bryant. A mouse hover allows an authorized user to see individual student names at points of interest, in this case, a student whose score increased dramatically in 2009 compared to 2008.

A simple survey instrument was created, asking users to compare this system to ADE's current longitudinal data system. The visualization tool and framework were presented to groups of educators in different parts of the state. Data will continue to be collected as more educators are shown the system, and feedback from their use will be used to make modifications.

RESULTS

Even though hive and the data analysis framework are very new, the preliminary data is very encouraging. We have had 200 responses to the survey to date, and 95% agree or strongly agree that the tools represent a clear process for analyzing data which they did not have before. From all respondents, 91% report that the tools provide a clear process for identifying and implementing needed changes based on data analysis. Also, 94% of respondents report that the tools will help them look at data objectively while avoiding the bias of prior knowledge. Finally, 97% report that the system will allow them to collaborate with colleagues outside their school or district about data.

CONCLUSIONS

Preliminary data show that users strongly believe the new visualization tool and data analysis framework is an improvement over the existing longitudinal data system. As mentioned before, ADE's existing system was already held in high regard, so it is very

significant that respondents are finding the addition of hive and the framework to the existing system as much better than what was already seen as an excellent data system. By researching how users interact with the data and the common mistakes they make in data analysis, we have taken a strong system and made it even stronger. We believe all information products should undergo similar research to ensure that such products truly meet the "fitness for use" criteria and that the interaction between user and data be part of any information quality plan. Current research in information quality focuses mainly on data inputs, but this research shows that a product that is seen as exhibiting high quality can be made even better by focusing on consumers use of the data.

One area of hive that has not been heavily used is the social networking aspect of the system. While users have created many visualizations, there are very few threaded discussions. We are currently building group functionality into the social networking tool, where, for example, a teacher can join a group of fourth grade literacy teachers and be notified when a new visualization is created using that data. It remains to be seen if educators will embrace social networking tools as part of their data analysis, but we believe online collaboration to be an important part in ensuring that Arkansas' educators get the best use out of these data systems.

REFERENCES

[1 National Center for Education Statistics: Institute of Education Sciences, Statewide longitudinal data systems grant program, http://nces.ed.gov/programs/slds/ Retrieved May 24, 2009.

[2] National Center for Educational Statistics, Forum guide to decision support systems, http://nces.ed.gov: http://nces.ed.gov/pubsearch retrieved May 24, 2009.

[3] Marsh, J. A., Pane, J. F., & Hamilton, L. S. (2006). *Making Sense of Data-Driven Decision Makingin Education: Evidence from Recent RAND Research*. http://www.rand.org/pubs/occasional_papers/2006/RAND_OP170.pdf retrieved May 16, 2009

[4] Boudett, K. P., City, E. A., & Mumane, R. J. The "Data Wise" improvement process, *Harvard Education Letter*, 22, (1) 1-3, 2006, http://www.hepg.org/document/1/ retrieved May 14, 2009.

[5] Bernhardt, V. L., & Geise, B. J., Data, data everywhere: ASCD annual conference, 2009, http://eff.csuchico.edu/Downloads/ASCD%202009/ASCD%20Notes%20Mar09.pdf retrieved May 14, 2009.

[6] Mandinach, E. B., Honey, M., Light, D. (April 9, 2006). A theoretical framework for data-driven decision making, http://cct.edc.org/admin/publications/speeches/DataFrame_AERA06.pdf retrieved October 6, 2008.

[7] Ikemoto, G., Marsh, J., Cutting through the "data-driven" mantra, http://www.rand.org/pubs/reprints/2009/RAND_RP1372.pdf retrieved October 6, 2008.

[8] Simon, H., Designing organization for an information-rich world, 37-72, *Computers, communications, and the public interest,* Baltimore, MD: The Johns Hopkins Press, 1971.

[9] Stanovich, K., The fundamental computational biases of human cognition, 291-342, *The Psychology of Problem Solving* New York, Cambridge University Press, 2003.

[10] U.S. Chamber of Commerce, Leaders and Laggards, http://www.uschamber.com/reportcard/default, retrieved November 3, 2009

[11] Van Houten, L., Miyasaka, J., Agullard, K., Zimmerman, J., *Developing an Effective School Plan: An Activity-Based Guide to Understanding Your School and Improving Student Outcomes*, Oakland, CA, WestED, 2006

[12] Heer, J., Card, S., Landay, J., Prefuse Information Visualization Toolkit, http://prefuse.org/, retrieved March 3, 2008.

[13] Howe, J., *Crowdsourcing: Why the Power of the Crowd is Driving the Future of Business*, New York, Crown Business, 2008.

CS0: WHY, WHAT, AND HOW?*

PANEL DISCUSSION

Matt Brown
Arkansas Tech University
Russellville, AR
hbrown11@atu.edu

Chenyi Hu
University of Central Arkansas,
Conway, AR
chu@uca.edu

Carl Burch
Hendrix College, Conway, AR
cburch@cburch.com.

Michael Nooner
University of Central Arkansas,
Conway, AR
mnooner@uca.edu

CS1 has been commonly offered as the first introductory course in Computer Science. For various reasons, the panelists have developed and offered a CS0 course at different institutions during the last several years. The panelists will share their experience in designing and implementing a CS0 course at their institutions. Discussions will follow on the following questions:

1. Why do we need a CS0 course? Whom should this course be for?
2. What should be the expected objectives and appropriate contents of the course?
3. How do we design, implement, and assess such a course?

POSITION STATEMENTS

Matt Brown, Arkansas Tech University

Students desiring to major and work in the field of computing often begin their college career with many questions and misconceptions. A CS0 is one avenue to aid both the student and university in addressing these issues. The course offered at Arkansas Tech is CS0, IS0, and IT0 in the sense that it serves as an orientation to the three different undergraduate fields of study, Computer Science, Information Systems, and Information Technology. The course is required for all prospective CS, IS, or IT majors. A primary goal is to offer students a chance to discern which of these three fields of study, if any, is right for them. Additional goals include an introduction to the systems that will be used in later courses, building problem solving skills, and providing students a broad and lively survey of the topics within the fields (i.e. the breadth-first approach). By touching on a diverse subject matter in CS0, students may find one or more topics that interest

them and subsequently may find motivation before beginning courses with narrow focus and challenging content.

Carl Burch, Hendrix College

In the environment of a liberal arts college, CS0 is generally most useful for students who are discovering whether computer science is for them: Those who find programming tedious can still complete the course without growing unduly frustrated by increasingly complex programs, and they can leave the course knowing fundamentals of the discipline that are unrelated to programming. Such a course can double as a service course for students from a variety of disciplines. But CS0 can be problematic in circumstances where other departments (such as physics or engineering) want their students to learn programming, and it is problematic when many students enter with AP or transfer computer science credits. Another challenge for CS0 is cohesion among topics.

Chenyi Hu and Michael Nooner, University of Central Arkansas

The knowledge of computing has evolved dramatically during last few decades. It is a challenge for both educators and students to effectively introduce and learn fundamental computer science. Many students are turned away from computer science even in the early stages of their education. The University of Central Arkansas is a comprehensive university serving about 12,000 students mostly from the State of Arkansas. As with many other public institutions, some students majoring in computer science do not have appropriate academic preparation. These students often have difficulties in CS I and II. Our initial motivation was to develop a CS0 course for remediation to help improve the retention rate. However, when we first offered the course in the spring of 2008, most students were non-CS majors. We realized that the CS0 course should also be designed and implemented for a broader range of students, to introduce computational thinking and problem solving. We will share our experience with participants.

ABOUT THE PANELISTS

Dr. Brown is an Assistant Professor of the Department of Computer Science at Arkansas Tech University (ATU). Before joining ATU in 2008, he held positions in programming, data mining, and statistics at the Wal-Mart Home Office and Tyson Foods World Headquarters over the course of nine years. He received his Ph. D. from Nova Southeastern University in 2007.

Dr. Burch is the Department Chair and Associate Professor of Computer Science at Hendrix College. Before joining Hendrix College in 2004, he was on the faculty at the College of Saint Benedict and Saint John's University, where he taught CS0 four times and co-authored a paper about that curriculum [1]. He received his Ph. D. in Computer Science from Carnegie Mellon University in 2000.

Dr. Hu is the Department Chairperson and Professor of Computer Science at the University of Central Arkansas (UCA). Before joining UCA in 2002, he had been on the

faculty of Computer and Mathematical Sciences at the University of Houston-Downtown for twelve years. He received his Ph. D. from the University of Louisiana at Lafayette in 1990. He taught CS0 at UCA twice.

Mr. Nooner is an instructor and the system administrator for the Computer Science Department at UCA. He received both of his B.S. and M.S. degrees in Computer Science from UCA. He taught a CS0 course three times at UCA.

REFERENCES

[1] Burch, C., Ziegler, L., "Science of Computing Suite (SOCS): Resources for a Breadth-First Introduction," *Technical Symposium on Computer Science Education* (SIGCSE), 437-441, 2004.

LABGRADER AND NIFTY ASSIGNMENTS*

TUTORIAL PRESENTATION

Bob Bradley and Paul Tesar
Department of Management, Marketing, Computer Science & Information SystemsThe
University of Tennessee at Martin
Martin, TN 38238
731 881 7238
bbradley@utm.edu

This tutorial will consist of a short presentation about LabGrader and its features, as well as a hands-on nifty assignment session.

LabGrader is a locally written web-based system that assists in the teaching and grading of introductory (C++, Java, etc.) programming classes. LabGrader allows students to enter, compile, test and debug their programming assignments all from the comfort of a web browser. It includes a web-based IDE, and students can submit their assignments and view their grades along with comments from the instructor. From a web browser, instructors can create new assignments, create new unit tests, view the grade-book and test and grade submitted assignments. These assignments can have multiple "unit" tests which the students run to ensure that their output is correct.

LabGrader is a free and open source system that instructors can download and use. It is written in Microsoft ASP.Net MVC and is designed to be run on a Windows 2008 server with a MySql back-end. LabGrader supports multiple teachers and classes, and it can be integrated with Active Directory.

The presentation will be followed by a hands-on session where attendees can try the LabGrader system. Attendees will be able to try some of the sample assignments or practice creating their own new assignments.

While most of the assignments will be of the text input/output type, this session also will present some nifty assignments that can be given with (or without) the LabGrader system. These nifty assignments include HTML assignments, photo image filter assignments and audio assignments.

NETWORK SECURITY IN TWO-YEAR COLLEGES*

Russell Jones
Computer and Information Technology Dept.
Arkansas State University – Main Campus
State University, AR 72467
(870) 972-3988
rjones@astate.edu

Tamya Jean Stallings
Information Technology Services
Arkansas State University – Newport
Newport, AR

ABSTRACT

Network security has become a growing challenge in industry and education. With the increase of viruses, spam, hackers, and identity theft, organizations are beginning to look at areas to improve the protection of their Information Technology structure. Higher Education has the highest vulnerability to network security threats, especially the two-year institutions. Computer labs within institutions of higher learning are often known for their open or ease of access. Open labs, limited staff and resources hinder security measures for these institutions. This paper focuses on the current state of network security at two-year institution.

Conflict, war and terrorism have elevated both apprehension in high technology industries and awareness about the need for improved physical security as well as information security (Casey, 2002).

First among the targets in cyberspace are colleges and universities because they possess a vast amount of computer power most of which contains resources with open access. Hackers attack educational institutions due to their perception of lax security that is coupled with high capacity computer systems. Research lab servers and workstations as well as student computers in dorms are not typically managed by IT staff thus opening those resources for hackers and viruses (Updegrove & Wilson, 2003). The National School Safety Center has acknowledged these open resources and is in the process of

ensuring that higher educational institutions become more secure (National Strategy, 2003).

Higher education institutions have a moral as well as legal responsibility to protect the sensitive data maintained on campus computers. The computers contain records with financial, medical, and student data, intellectual property and digital communications both internally and externally that are used in day-to-day activities (Oblinger, 2003). A balancing act between cyber-security and the principles of the academy must be struck so that institutional resources are protected without compromising the institution's mission.

Despite the recommendations from NSSC, academia has typically treated Information Security as an after-thought. It is an area that has been left for the Information Technology departments to worry over and pay for out of their already thinly stretched budgets. For years it has been the responsibility of overburdened system administrators or general IT staffers to make sure that the campus was safe and secure (Recor, 2003). IT staff are faced with the daunting task of securing data while allowing an open and convenient environment in academia. The balancing act between convenience, openness and security measures is the biggest challenge for academia. Campus policies and procedures must have the hierarchical support in place for a successful and organized security infrastructure (Recor, 2003).

Two-year colleges run into a plethora of barriers when it comes to the implementation of network security. A large barrier for any organization would be the lack of financial resources. IT management must take numerous items into consideration for network security which can become very expensive. Some considerations should include physical aspects, hardware, and software. Staffing the IT department becomes a barrier after the purchase of the hardware and software. An institution may have hardware and software purchased for network security but the IT staff may not have the skills necessary or the time to install and maintain the resources. When management does approve funding for the equipment the next barrier would be security training for the installation of equipment to make it work efficiently (Daniels, 2001).

Though network security has been heavily investigated in the for-profit and four-year college sectors, very little research has been conducted of this issue in two-year colleges. The study will fill this gap in knowledge by surveying the stakeholders in this area thereby providing IT professionals with a report on the status of security in community colleges. The results will allow community college IT professionals to compare their environment to others in the region so that they can advocate for services that are appropriate for two-year campuses.

METHODOLOGY

The purpose of this study was to determine how two-year colleges have protected their campus computer networks from viruses and hackers. With the increase of viruses and computer hackers, it is important to know whether the campuses have been proactive with their security or reactive based on how their campuses have been affected.

To accomplish the study's purpose, a survey was sent to all community colleges in Arkansas and its six neighboring states in order to investigate the emphasis that two-year

colleges have placed on network security. The primary research question was to determine the status of network security in community colleges in Arkansas and the six surrounding states.

The Director of Computer Services was the targeted respondent for this descriptive study designed to address the concerns directly relating to network security and how policies and procedures are affected. The Directors were asked to complete the questionnaire since they typically had responsibility for community college computer networks in general and would be most likely to respond. It was assumed that they had the most readily available and accurate information needed to complete the questionnaire.

RESULTS

Forty-five of the 134 surveys sent to the other six states were returned. The study had a 33.6% questionnaire return rate. The respondent was asked if their organization was where it should be in various areas of network security. Each respondent was asked if, in their opinion, the organization was where it should be with network security. Over 74% stated their organization was not where it should be with network security. The overwhelming explanations of what this existed was: lack of funding (29%), lack of staff (26%), lack of hardware (16%) and lack of software (18%). Some additional reasons why institutions were not where they should be included lack of planning, training, time, not a priority and security has not been a problem, and upper management had a lack of understanding. Respondents were also asked to indicate whether certain security procedures were in place at their institution and if they thought they were necessary to provide an adequate level of security. These responses are summarized in Table 1 below.

Security Issue	% Currently in Place	% Thought Was Needed
Strong Passwords	50%	93%
Require Password Changes	64%	89%
Multiple Level Authentication	65%	84%
Regular Risk Analysis Performed	34%	86%
Firewalls on External Connection	94%	93%
Internal Firewalls	64%	80%
Encrypted Files Used	31%	63%
Routine Backups Performed	94%	100%
Security Patches Regularly Applied	86%	98%
Virus Protection Routinely Updated	96%	100%
Intrusion Detection Systems Utilized	40%	91%
PKI Certificates Utilized	16%	31%
Smart Cards Utilized	6%	28%

Respondents were also asked how the institutions ensure their wireless connections are secure if they have that option. Respondents were asked to list ways that wireless connections were secured. The majority of the responses included the following:

- use Blue Socket Appliance
- limit access by using MAC addresses
- don't allow students access to wireless at this time
- firewalls on access points
- require authentication
- virtual private networks
- WPA/WPA2, 802.1x, WEP
- VLANS

Finally, they were also asked about the use of patch servers. These were implemented at 38 (75.0%) of the institutions. The 25.0% that did not have a patch server were asked how the computers at the institution were updated. A list of three items was provided with an area for additional responses to be added. Multiple answers could be selected by the respondents for this question. Seven of the respondents who reported they didn't have patch servers indicated the IT staff visits each computer. Four participants said that the individual employees were responsible. Three institutions set the computers to automatically update the computers.

DISCUSSIONS, IMPLICATIONS, AND CONCLUSIONS

Network security has become a growing challenge in academic institutions. Higher education institutions are beginning to look at areas to improve the protection of their Information Technology structure due to the increase of viruses, spam, hackers, and identity theft. Educational institutions have the highest vulnerability to network security threats.

The Information Technology profession in the community college setting has acknowledged the importance of security, both physical and network. The focus on this subject gains strength every time a new spam technique is released or when a new disaster like Katrina hits. The study indicates the core beliefs and prioritizing of Information Technology personnel regarding security.

It is valuable to know what procedures are being encouraged by the leadership of the IT departments in two-year colleges. The data indicates that directors have a real concern about the time and money demands, administrative positions and the stressful nature of the jobs.

At two-year colleges, 72.5% of Informational Technology staff spends about 0-20% of their workday dealing with security issues. These issues include updating software, monitoring firewalls, monitoring viruses, monitoring network activity and other similar measures. The time spent on security is minimal considering all the hats that the community colleges IT staff wear. Only 25.5% of the institutions included in the study have a security support specialist on staff. The majority of these institutions (41.2%) do not have anyone who has had more than a year of security training. The data suggests that IT directors want to increase the knowledge and skills available to the IT departments

but currently are relying on individual staff member's skills to react to security breaches. The study supports the finding of the Educause Core Data Service 2003 Summary Report (Educause Core Data Service, 2004). The lack of training is due to numerous factors such as lack of funding, lack of staff, and lack of senior administrative support. Administration has to get behind the IT staff and find the resources needed to get the security expertise on their campus. Payne (2003) discussed having a security training tailored for the administrators due to their decision making for allocating resources. Network security should be marketed in business terms to the administrators to show how the institution as a whole can be affected by security breaches (Payne, 2003). Another article states that a computer service department does not have any protection, creditability, or funding for security initiatives if the management does not get behind the department (Hayes, 2001).

It is concluded that two-year colleges are facing a real crisis in IT security practices. Many administrators are already addressing these issues but at a tragically slow rate. The majority of current administrators have reported a lack of support from other administrative positions and barriers such as excessive time requirements and stress which may prevent them from incorporating policies that would more securely protect their campus network.

REFERENCES

Casey, T. (2002, August 29). *Option 1 – research on topics in information security the national strategy to secure cyberspace: An in-depth review*. Retrieved January 7, 2005 from http://www.giac.org/practical/GSEC/Tim_Casey_GSEC.pdf.

Daniels, Jack. (2001). *The weakest link…This is not a game*. Retrieved June 16, 2006 from http://www.sans.org/readingroom/whitepapers/basics/440.php.

Educause (n.d.b). *About educause*. Retrieved June 24, 2006 from http://www.educause.edu/about.

Educause Core Data Service. (2004). 2003 summary report. Washington, D.C.:Hawkins, B.L., Rudy, J.A.; & Madsen, J.W.

Hayes, H. (2001, June). Security training checklist. Retrieved June 16, 2006 from http://www.sans.org/readingroom/whitepapers/basics/440.php

National Strategy to Secure Cyberspace. [www.securecyberspace.gov].2003.

Oblinger, D. (2003). IT security and academic values. In M. Luker & R. Peterson (Eds.) Educause leadership strategies: Vol. 8. *Computer and network security in higher education* (pp.31-44). San Francisco, CA: Jossey-Bass.

Payne, S. (2003). Campuswide security education and awareness. In M. Luker & R. Peterson (Eds.) Educause leadership strategies: Vol. 8. *Computer and network security in higher education* (pp.31-44). San Francisco, CA: Jossey-Bass.

Recor, J. (2003). Organizing for improved security. In M. Luker & R. Peterson (Eds.) Educause leadership strategies:Vol 8 Computers and network security in higher education (pp15). San Francisco, CA:Jossey-Bass.

Updegrove, D. & Wishon, G. Foreword.. In M. Luker & R. Peterson (Eds.) Educause leadership strategies: Vol. 8. *Computer and network security in higher education* (pp.xi-xiv). San Francisco, CA: Jossey-Bass.

TEACHING FEDERATED IDENTITY IN COMPUTER AND INFORMATION SCIENCE*

George Whitson
Computer Science Department
The University of Texas at Tyler
Tyler, Texas 75799
903-566-7006
gwhitson@mail.uttyl.edu

ABSTRACT

Computer security has become a major area of computer science. Most colleges and universities now teach several courses in computer security and also cover security topics in many of their courses. The introductory computer security course covers security mechanisms like encryption, message authentication techniques and SSL. The introductory programming course now discusses how to avoid buffer overflow attacks rather than just looking at data validation. But, Federated Identity, a logical extension of authentication, is often little mentioned outside of Web security classes. In this paper we describe Federated Identity as it is currently practiced and then describe a number of courses where we have covered both the theory and practice of Federated Identity.

INTRODUCTION

When accessing a computer a user must establish an identity. Initially, the user may login to a workstation on a corporate network after being authenticated by a central server using a CHAP login/password process. In addition, authentication may require the user to have a digital certificate on a flash-drive, possess a secret piece of information or demonstrate a biometric trait. After the authentication is finished, the identity then becomes a list of attribute/value pairs that uniquely identify the user in the domain of the authentication server.

If the user needs to access resources in another authentication domain, then they need to establish an identity in that domain as well. Multi-domain authentication was introduced by Microsoft in the Windows NT operating system [12] and by OSF's Kerberos by in the 1980's [10]. Microsoft's original multi-domain authentication has been replaced by Active Directory, and Kerberos has been implemented by many computer vendors. While both Active Directory and Kerberos support multi-domain authentication for a single vendor, neither of these systems currently supports multi-domain authentication across vendors.

Federated Identity refers to a user being able to have a simple sign-on in a multi-domain environment that is easy to use, provides adequate security and insures sufficient privacy.

BASICS OF FEDERATED IDENTITY

Cameron defines digital identity as a set of claims made by one digital subject about itself or another digital subject [3]. The Liberty Alliance defines (digital) identity as consisting of traits, attributes, and preferences upon which one may receive personalized services [7]. Combining these definitions we see that a digital identity consists of a set of attribute/value pairs that uniquely define a subject in a domain.

As Web services matured, the development of a digital identity for applications using Web services became important. In particular, for an application to function seamlessly as it accesses remote Web services, each user of the application needs a Single Sign-On (SSO). Standards have been developed by W3C and OASIS to support SSO for Web Services [9, 16]. These SSO standards are actually quite general, and a number of organizations are modifying them to provide an architecture for true Federated Identity [1, 4, 13]. Microsoft announced Geneva as its Federated identity product and recently changed the name to Windows Identity Foundation (WIF) [2].The Liberty Alliance is also developing a Federated Identity standard called the Identity Architecture [7].

Kim Cameron, of Microsoft, has listed seven rules that any Federated Identity architecture should satisfy [3]. These are generally recognized as a good starting point of a discussion of Federated Identity:

1. **User Control and Consent**. Federated identity systems must only reveal information identifying a user with the user's consent.
2. **Minimal Disclosure for a Constrained Use.** The solution which discloses the least amount of identifying information is best.
3. **Justifiable Parties**. Digital identity must be designed so that the disclosure of identifying information is limited to authorized parties.
4. **Directed Identity.** A universal identity system must support both "omni-directional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities.
5. **Pluralism of Technologies.** A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers.

6. **Human Integration.** The universal identity metasystem must define the user to be a component of the distributed system and provide protection against identity attacks.
7. **Consistent Experience.** The unifying identity metasystem must guarantee its users a simple, consistent experience.

The first rule captures the essence of WIF. The WIF authentication process requires the user and the server to make a decision to share their authentication information. By proper signing of this information, the receiver can be assured of its authenticity. WIF is a user-centric Federated Identity architecture [2].

As with any active research area there are several groups proposing standards. Each group has an active research program that is producing a multitude of ideas. One of the standards is WS-Federation [8], and it provides a good starting point for classifying Federated Identity architectures. The three Federated Identity architectures described by WS-Federation are:

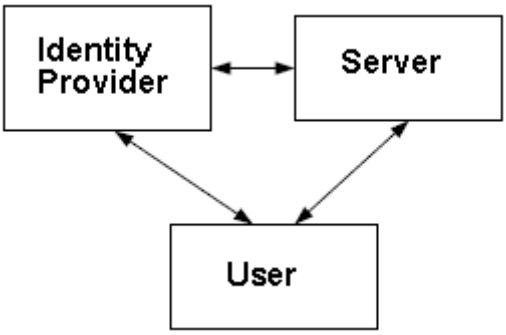
1. A digital identity is fixed across domains. Here a subject would only need to convince a provider of their identity which might be easily done by using a signed set of authentication credentials provides by a Security Token Service (STS).
2. There would be a pair-wise mapping of digital identities, where a unique digital identity is used for each principal at each target service. Here a subject would only need to map their identity to that maintained by each provider. Again, this might be easily accomplished by providing a STS security token to the provider, but might also require a trust relationship among the STS's.
3. Each service provider would have to generate a digital identity from the information provided by a subject, given a small amount of information about the subject. Again this would be accomplished by providing a STS security token to the provider and letting the provider generate the identity from a trust relationship it has with the STS's of the subject.

A SURVEY OF FEDERATED IDENTITY SYSTEMS USED TODAY

Cameron's seven rules and WS-Federation's basic architectures provide a foundation for Federated Identity systems. But a survey of current systems [5] being developed is not quite that neat. Looking at the current systems a reasonable practical classification of Federated Identity systems is:

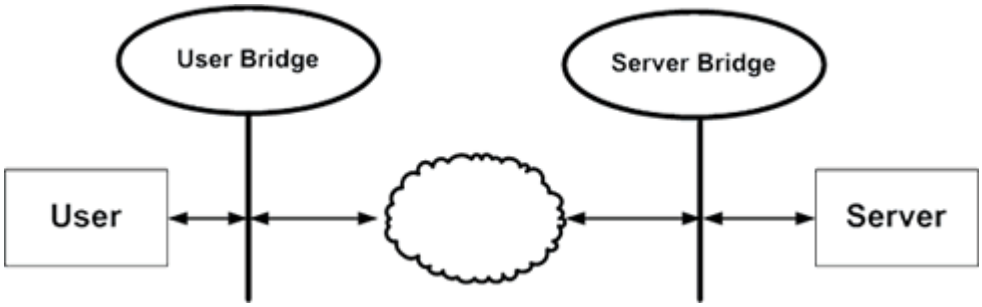
1. User centered – user/server exercises the ultimate control (CardSpace/InfoCard).
2. Bridge centered – user/server may connect to an intermediate authentication system (NCPDP and SAFE bioPharma).
3. Server centered - user/server identifier by designated server (Open ID).

Microsoft, IBM and the Liberty Alliance all appear to be supporting user-centric systems based on WS-Security [1]. Microsoft is centering its systems on SAML1 and WS-Federation, while the Liberty Alliance is centering its system on SAML2. As illustrated below,



A user requests a signed authentication pack from a trusted identity provider, sometimes after checking to see what credentials the server requires, and then sends them to the server to be authenticated by the server. While there can be substantial automation of all of these processes, the user and server have the control to either make a connection or to refuse it.

Some Federated identity systems add some administrative control to the authentication process. A good example of this type of Federated Identity system can be seen in a standard implementation of the NCPDP (National Council for Prescription Drug Programs) system for e-prescribing. Here users authenticate to a trusted bridge, the server also authenticates to a trusted bridge, and the user is authenticated to the server with the support of trusted bridges [11]. For NCPDP the diagram below illustrates the typical authentication.



In the current Safe bioPharma standard of the medical research community [14], users generally authenticate to a trusted bridge and the servers then accept the authentication of the trusted bridge, although more complex bridge structures are allowed. The SAFE in SAFE bioPharma stands for "Signatures and Authentication For Everyone." Thus, the bridge centered Federated Identity systems are similar to (and sometimes an extension of) the Certificate Authorities of PKI.

Microsoft's Passport, now called Windows Live ID, is a server-centric Federated Identity system. Here the user places their authentication information on an authentication

server and can either use a signed token from the authentication server or request that others contact the authentication server for authentication of the user. Other companies than Microsoft offer a SSO for the Web, but none of these have been widely accepted yet. An interesting server-centric Federated Identity system that has been well received is the OpenID standard [15]. It is a lightweight service to provide a single digital identity for a user to access all of the interactive Web applications, like FaceBook, that have proliferated over the internet.

TEACHING FEDERATED IDENTITY IN AN ADVANCED VISUAL BASIC COURSE

We began to cover computer security concepts in our Advance Visual Basic programming course several years ago. The main emphasis was on checking all user input, being sure to gracefully exit from program errors and threat analysis. In the fall of 2009, we added coverage of authentication for applications and how Federated identity could play a role in developing secure enterprise applications. Some details of what was covered in the course are:

1. As soon as students were introduced to Forms, they were also introduced to checking all user input for attacks, as an enhanced validating user technique.
2. Shortly before Midterm, I gave a set of lectures on computer and application programming security that stressed Howard's Ten Rules and Threat Modeling [16].
3. Early in the course, students were assigned a multi-Form program that combined several programs used to calculate reverse mortgages. After the security lecture, students had to return to this program and describe what changes were needed to make it more secure.
4. Later, a part of the program was implemented as a DLL. As a part of the security analysis, we discussed the problems of authenticating the information returned from a DLL. We then added a discussion of authentication if the contents of the DLL were accessed through COM or DCOM. This led to a thorough discussion of Federated Identity and its importance for application programming.
5. During the last two weeks of the course, we discussed how to implement a Web service that returned the mortgage data to a desktop VB application. A fairly complete overview of how WCF and Geneva work together to implement SSO for Web services was included with the material on Web services.

The introduction of Federated Identity, as represented by a SSO Web service application, was an ambitious project for a junior level programming course, but at least 60% of the students seemed to master the material, and 100% of the students appreciated covering some of the more recent topics in VB programming.

TEACHING FEDERATED IDENTITY IN WEB DEVELOPMENT COURSES

We teach a number of courses on building Web applications. Some stress Web design, some stress e-commerce and some stress building database-driven Websites. Web services, and the need to use Federated Identity, are now introduced in all of these courses. For example, in the Web Design for E-Commerce course, we always have a

chapter on computer security and secure payment methods. In the spring of 2009, we added lectures on the use of Web services in modern Web applications and then gave an overview of WS-Security, including material on how SAML authentication could be implemented for the class project. We gave demonstrations in class of how to authenticate in a single domain and had students write papers on how they would add multi-domain authentication for a Microsoft only environment.

TEACHING FEDERATED IDENTITY IN A MEDICAL INFORMATICS COURSE

In the spring of 2009, I taught a course on Medical Informatics. The main emphasis in the course was HL7 as a technology to support an Electronic Health Record. At the end of the class we added a detailed discussion of HL7's new authentication classes and the techniques for establishing a digital identity for an Electron Health Record. We also added a detailed discussion of the NCPDP standard for e-prescribing. We were able to get all three of the major hospital/clinic systems in our area to present an overview of their information systems, and each added a complete discussion of how they were approaching multi-domain authentication, Each of the hospital/clinic systems had contracted with a vendor to provide SSO service for their systems, that followed a user-centric approach to authentication. While we did not cover the theory of Federated Identity in detail in the course, the industry presentations provided an excellent introduction for students to some applications of Federated Identity.

TEACHING FEDERATED IDENTITY IN COMPUTER SECURITY COURSES

We teach three computer security courses and cover Federated Identity in all of these. In our basic security course, we have an extensive coverage of authentication for workstations, network domains and the Web. We then look at SSO for Web services for a week, including a brief introduction to WS-Security. Students are then assigned two papers to write about Federated Identity. The first is a guided discussion of the general Federated Identity architectures, similar to that in this paper, and the second is a detailed description of one real Federated Identity system, like Microsoft's WIF. Students also set up an Active Directory certificate authentication system for Windows Server 2003, and we discuss how it can be a part of a multi-domain Federated Identity system based on certificates.

In our Computer Security Management course, students are formed into teams which plan and develop a certification document (including a plan, risk analysis and policy) for a real system we set up in our security lab. This certification includes a Federated Identity component. The teams are paired so that after they complete the certification of their systems, including the need to support Federated Identity, each team does an accreditation of another team's certification.

In our Web security course we give a complete coverage of basic authentication, including CHAP, Kerberos and certificates. We also cover TLS, as a way of authenticating Web users and servers, as well as supporting secure transmission. We then give a detailed coverage of the literature on Federated Identity [9, 16]. With this

background we then cover Web services security in detail [9, 16]. Finally, we finish the course by applying the techniques of Web service security to the general security needed in Federated Identity. In this course, students implement a simple multi-domain Web service example with SAML authentication using certificates. Both domains are currently Windows domains, but multi-domain authentication across different operating systems is planned for the future.

SUMMARY

Federated Identity is an important topic in computer security and is now covered in all of our security courses, and many other courses as well. This is a complicated subject and can only be introduced in most courses. Developing multi-domain applications is still hard, so in most of our courses at the present time, we cover only the theory or provide demonstrations. In our Web security we do a thorough job covering the theory, and students implement a simple multi-domain example using Microsoft's technology. As the Federated Identity architectures become more robust we plan to introduce multi-vendor, multi-domain examples into all of our courses.

REFERENCES

[1] Balasubramaniam, Lewis, Morris, Simanta, Smith, Identity management and its impact on federation in a system-of-systems context, *3rd Annual IEEE Systems Conference,* (March 2009), p179-182.

[2] Bustamante, Michele, Claims-Based Apps: Claims-Based Authorization with Windows Identity Foundation, *MSDN Magazine*, November 2009(Vol. 24 Number 11), p 34-47.

[3] Cameron, Kim, The Laws if Identity, http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf retrieved October 11, 2009.

[4] Fragoso-Rodriguez, et al., Federated Identity Architectures, *Proc. 1st Mexican Conference on Informatics Security, 2006 (MCIS'2006)*, Oaxaca, Mexico, IEEE Computer Society, Mexico, p 1-8.

[5] Goth, Greg, Identity Management, Access Specs Are Rolling Along, *IEEE Internet Computing*, Volume 9, Issue 1, Jan-Feb 2005, p 9-11.

[6] Howard and LeBlanc, *Writing Secure Code, Second Edition*, Microsoft Press, 2002.

[7] Liberty Alliance, Introduction to the Liberty Alliance Identity, http://www.projectliberty.org , retrieved October 11, 2009.

[8] Lockhart, Hal, et al., Web Services Federation Language (WS-Federation), Version 1.1(December 2006), http://www.ibm.com/developerworks/library/specification/ws-fed/, retrieved October 11, 2009.

[9] O'Neill, Mark, et al., *Web Services Security,* McGraw-Hill (Osborne), 2003, ISBN 0-07-222471-1.

[10] Rosenberry, Kenney, Fisher, *Understanding DCE,* O'Reilly Media(1992).

[11] Rothermich, Swanson, Electronic Prescribing Security and Authentication, http://www.ncvhs.hhs.gov/041209p1.pdf, retrieved October 11, 2009.

[12] Russinovich, Solomon, Solomon, Allchin, *Inside Microsoft Windows Internals, Covering Windows 2000, Windows XP, Windows Server 2003,* Microsoft Press, 2005.

[13] Shim, Bhalla, Pendyala, Federated Identity Management, *IEEE Computer*, Volume 38, Issue 12, Dec. 2005, p 120-122.

[14] Stelex, Meeting the Need for a Global Identity Management System in the Life Sciences, http://www.safe-biopharma.org/infocenter/a-safewhitepaperstelexfinal.pdf**,** retrieved October 11, 2009.

[15] Weitzner, D.J., Whose Name Is It, Anyway? Decentralized Identity Systems on the Web, *IEEE Internet Computing*, Volume: 11, Issue: 4, July-Aug. 2000, p 72-76.

[16] Whitson, G., Security for Service Oriented Architectures, *Journal of Computing Sciences in Colleges*, Volume 23, Issue 4(April 2008) p 8-9.

THE PNTFS FILE SYSTEM IN THE MOSES2 OPERATING SYSTEM ENVIRONMENT SIMULATOR*

Robert England
Transylvania University
300 North Broadway
Lexington, KY 40508
Ph: 859-576-7021
rengland@transy.edu

ABSTRACT

The Moses2 operating system environment simulator is a software tool developed by the author to teach concepts of operating systems in upper level undergraduate O/S courses. Using the Moses2 system, students write their own version of a limited functionality operating system kernel that manages the processes and resources that are provided with the Moses2 simulator. This paper describes the PNTFS file system component that students develop as one of an incremental series of Moses2 features.

OVERVIEW OF THE MOSES2 SYSTEM

The Moses2 operating system environment simulator is a software system that provides a platform consisting of simulated hardware and a set of simulated user processes for student O/S construction projects in upper level undergraduate operating systems courses. Using Moses2, students design and implement their own small operating system kernel, which runs Moses2 user processes on the Moses2 simulated hardware. The Moses2 simulated hardware includes a Program Status Word, or PSW, a set of general purpose registers, a main memory area, a virtual output device, and a virtual disk drive. All of the simulated user processes that are provided as part of the Moses2 system are specially constructed to run only on the Moses2 simulated hardware and to request services that require the use of Moses2 simulated resources.

The conceptual and implementation details of the Moses2 virtual hardware machine, the user process model and context switching among processes, support for virtual memory, and mechanisms for shared resource management have all been previously described in [2,3,4]. After students have incrementally incorporated these features into their operating system kernel programs, the next in the sequence of Moses2 O/S projects that students undertake is the design and implementation of a simulated file system, which is described here.

GOALS OF THE FILE SYSTEM PROJECT

Clearly, with any operating system simulation system designed for use in a college course, the primary challenge is how to clearly illustrate both the types of problems that must be solved and the types of services that must be provided by an operating system, but while still keeping the scope of O/S code development within the realm of what can reasonably completed in one or two college semesters. The development of the Moses2 simulation system itself has proceeded since its beginning with this challenge and this constraint as its primary and guiding concerns.

The main goal of the PNTFS Moses2 file system project is to help students appreciate and understand the inner workings of a real file system, but without asking students to actually implement software that is as extraordinarily complex and lengthy as that of a real file system. In the PNTFS assignment, students implement a user-level API to map four high-level file operations --- open, close, write, and print --- to underlying low-level operations provided by the Moses2 simulator, such as disk cluster allocation. The hands-on experience achieved by implementing this simulated file system, limited though it may be, is invaluable to illustrating the types of problems that must be solved in the design and implementation of a real file system.

Furthermore, writing a Moses2 file system gives students the chance to see and develop real, practical applications of computer science course material that they may have viewed as more abstract than practical in their other courses such as data structures and systems programming. For example, following the model of the Windows NTFS file system, students are encouraged to implement their file system indexes for PNTFS as B+ trees. These trees are themselves stored externally among the set of virtual clusters on disk, providing extra programming practice for students in managing a large nontrivial data structure. Also, the writing to and reading from the file buffers which students must manage in the file handling component of their O/S requires treating the memory frames involved in these operations as bounded buffers: a user process may write varying numbers of characters at a time to an output file, for example, and the student O/S file manager is responsible for recognizing when a virtual frame in memory has become full, and therefore must be flushed out to disk.

Thus, the experience of constructing a PNTFS file system for use in the Moses2 simulation environment can serve the dual purpose of solidifying the basic concepts of O/S file services while avoiding intractable complexity, as well as providing a challenging test of students' advanced software construction skills.

THE PNTFS FILE SYSTEM ENVIRONMENT

The Moses2 file system name PNTFS (for PseudoNTFS) was suggested by students in an operating systems class because the system is conceptually based loosely on the design of the Windows file system NTFS [1,5]. Some features of the NTFS system, for example, the B+ tree structure model for the file index data, are used in PNTFS in a simplified form.

Before beginning to implement the PNTFS file system in a Moses2 project, a student must first have a fully functioning Moses2 operating system implemented through the level of all of the earlier incremental Moses2 assignments. In other words, context switching among processes must be in place, virtual memory must be implemented, and so on, through all features described in [2,3,4].

The descriptions of the technical details about the file system environment that are described here are offered primarily to relate this project back to other Moses2 O/S projects covered in the earlier papers, and to illustrate the level of technical detail that students must grapple with as they develop their own O/S kernel projects. New hardware and software specifications developed specifically for the file system project include the following:

- The cluster size, or disk file read/write unit, is the same as a virtual page size: 64 bytes.
- A total of 4096 virtual disk clusters are available on the virtual disk drive for use by the student O/S file system and for storing user files, starting at a virtual disk address called *cl_base* which is provided to the student O/S in simulated register Tx at system startup.
- The following new system-level functions are provided as part of the Moses2 simulator:
    - void frPrint (unsigned frame_no); // frame print

      The frPrint function prints the current contents of virtual memory frame number *frame_no* to the screen as a plain text string. This function is a valuable tool for debugging.
    - int clAlloc (unsigned cluster_count); // cluster allocate

      The clAlloc function allocates a total of *cluster_count* clusters from the virtual disk, if that many are available, and returns 1 or 0 to indicate success or failure. The first cluster allocated during the first run of a student O/S will always be cluster 0, so the student O/S can be written to assume that it can load its saved file system information by starting with cluster 0.

When a call to clAlloc is successful, the student O/S will find in register Sx the address of an array of 32-bit unsigned values that contain the following:

0th element: 0th starting logical cluster number
1st element: number of clusters in the 0th run
2nd element: 1st starting logical cluster number
3rd element: number of clusters in the 1st run
Etc.

The first negative value in an even-numbered array element marks the end of the cluster allocation. For example, if a Moses2 O/S successfully calls clAlloc requesting 10 clusters, then on return from clAlloc the O/S could possibly find the following array at Sx: 13 3 5 2 17 5 -1. This indicates that the system has claimed clusters 13, 14, 15, and 5, 6, and 17, 18, 19, 20, 21, for a total of 10 clusters, which the student O/S can now access through the pageRead / pageWrite interface for virtual disk reads and writes. This interface is described in detail in an earlier paper on virtual memory in Moses2 [2]. Contents of the virtual disk in Moses2 are automatically preserved across runs of a student O/S program, so the file system can be restored in its entirety to the state that it was in at the end of the previous run of the system.

Rather than simply reading from or writing to a file using standard input and output operations, a user process in Moses2 only accesses files through a well-defined interface of four system calls:

- System call 4 Activity 0 opens a file
- System call 4 Activity 1 closes a file
- System call 4 Activity 2 writes to an open file
- System call 4 Activity 3 prints the complete contents of an open file to the terminal screen

The occurrence of any one of these calls is communicated to the student O/S though a set of bits in the simulated PSW that are designated for just such system call communications. The simulated general registers communicate other information required per file operation, such as a 1 or a 2 from the user process in register Rx for Activity 0 to indicate whether the file should be opened for reading or writing, and a unique "file handle" integer (determined by the file naming scheme set up by the student O/S developer) returned to the user process by the O/S in register Sx after a file is opened successfully. Other information communicated via the registers for these system calls includes sequences of bytes to be written to a file, the count of bytes to write, and the file handle of a file that a user process wants to close. Return values from the O/S in register Rx report back to the user process whether or not the requested file operation completed successfully.

STUDENT IMPLEMENTATIONS OF PNTFS

Subject to the requirements and constraints imposed by the Moses2 simulator and described above, the student implementers of a PNTFS system are free to design and implement any type of file system they choose, as long as the required functionality is complete and correct. Most students so far have chosen to model their file systems after the NTFS system, since that was the model that they felt most familiar with when the project was assigned.

CONCLUSIONS AND FUTURE WORK

The PNTFS project is the newest of the Moses2 operating system projects. To date, it has only been assigned as required course work in one special topics course offered by

request to computer science students who had already completed a standard operating course which was taught by the author. Consequently, these students had already written a Moses2 kernel of their own, complete with the requisite functionality of context switching, virtual memory, and so on, so only the file system component remained to be implemented. Results to report are thus largely anecdotal, but uniformly positive: Students felt that they did in fact have a much better understanding of file systems in general, and after taking standardized tests such as the GRE Computer Science Subject Test, they reported that they had been quite well prepared for the questions about operating systems and file systems.

The incorporation of a file system project into the suite of Moses2 operating system construction projects may well represent the final addition to this collection that can reasonably be completed by students in the time available to them during an operating systems course. The author plans to apply for an NSF CCLI grant in the spring of 2010 to complete the preparation of the Moses2 system for distribution to other schools so that any instructor who wishes to use the Moses2 system in an operating system course may freely do so.

REFERENCES

[1] Custer, H., *Inside the Windows NT File System*, Redmond, WA: Microsoft Press, 1994.

[2] England, R.E., Teaching concepts of virtual memory with the Moses2 microcomputer operating system environment simulator, *Journal of Computing Sciences in Colleges, 20*, (6), 84-91, 2005.

[3] England, R.E., Teaching principles of shared resource management with the Moses2 microcomputer operating system environment, *Journal of Computing Sciences in Colleges, 21*, (5), 46-52, 2006.

[4] England, R.E., The virtual machine and user process model used in Moses2: a microcomputer operating system environment simulator, *Journal of Computing Sciences in Colleges, 17*, (2), 301-309, 2001.

[5] Nagar, R., *Windows NT File System Internals: A Developer's Guide*, Sebastopol, CA: O'Reilly, 1997.

JAVAWIDE – INNOVATION IN AN ONLINE IDE*

TUTORIAL PRESENTATION

Jam Jenkins, Evelyn Brannock, Sonal Dekhane
Information Technology Program, Georgia Gwinnett College
(678) 407 – {5770, 5848, 5762}
{cjenkins, ebrannoc, sdekhane}@ggc.edu

ABSTRACT

Over the past decade applications have been moving from the desktop into the cloud. Only relatively recently have integrated development environments begun emerging in cloud computing. The Java Wiki Integrated Development Environment (JavaWIDE) is one of these new online IDEs, and it is designed to be simple enough for novice programmers to use. JavaWIDE is free and provides anyone the capability to create, edit and run programs anytime, anywhere, all from within a web browser – no specialized software required. JavaWIDE promotes collaboration, integrates well with social networking sites, and includes novel features such as concurrent editing support, a common code base for all users, revision history explorer, automatic posting of programs as applets, annotated and hyperlinked source code, an integrated Java API and many others. This workshop is presented by the developer of JavaWIDE Jam Jenkins, by Evelyn Brannock who has used JavaWIDE in the classroom, and by Sonal Dekhane who has surveyed students about their perceptions of using JavaWIDE.

INTRODUCTION

The objective of this tutorial/workshop is to give participants experience using JavaWIDE so that they can start realizing the potential of moving the IDE into the cloud. The developer will share his experience creating JavaWIDE and using it in the classroom for two years. Participants will use JavaWIDE during the workshop, and therefore should bring a laptop with wireless capability and Java 1.5 or higher installed (JRE or JDK). The target audience is those interested in cloud computing, innovative programming environments, collaboration, and those who teach introductory programming.

After briefly discussing the motivation for JavaWIDE and its development history, the majority of the workshop will be devoted to demonstrations and hands-on activities. This will include an introduction to concurrent editing and many other novel features,

using JavaWIDE with frameworks such as GridWorld, and handling file I/O securely. The workshop will conclude with some comments about the social networking aspects of JavaWIDE and student perceptions of using the system. Time at the end will be reserved to have some general discussion and address participant questions.

MOTIVATION

Setting up an efficient development environment is not always a simple task. The novice programmer can have an especially hard time learning, setting up and maintaining their own development environment. This simple observation motivated the creation of JavaWIDE.

Replicating the development environment to multiple platforms and different hardware is even more challenging. This is the problem faced by programming and software development instructors. Not only must they set up their own environment, but they also have to make sure all classroom computers are properly configured and they often must provide technical support to each student setting up his or her own environment on a variety of hardware and operating systems. Such activities are not only time consuming but they also require quite a bit of lead time for installation and testing.

One way to solve these problems is to avoid needing any specialized software at all – deliver the development environment using what is available on nearly every computer: a browser, an Internet connection, and the Java Runtime Environment (JRE) 1.5 or higher. Once the IDE moves online, many existing problems are eliminated and new opportunities emerge. JavaWIDE exemplifies what is possible as the IDE moves off the desktop and into the cloud.

DEVELOPMENT HISTORY

JavaWIDE was created in November 2007 as a basic Mediawiki extension. Since then it has grown in scope into a rich internet application that spans efficient client- and server-side scripts and programs. A high level overview of JavaWIDE's design will be described and justified with respect to security, performance, flexibility and scalability.

DEMONSTRATION

The demonstration portion of the workshop starts with creating, editing, and running an application and an applet. This demonstration includes important details such as how to handle the single namespace (because the single code base is shared), how compilation and runtime errors are detected and displayed, and how to use auto-import and code completion. After this brief demonstration, participants will be given time to write, edit and run their own programs.

TEACHING TIPS

Beyond being easy to use, JavaWIDE facilitates quite a few new teaching techniques based upon its novel capabilities. Since most have never used any concurrent editor

before, all workshop participants will be given this opportunity as they work together to create a single program at the same time. Following this experience, there will be a discussion about strategies for coordinating the editing process and for using concurrent editing to create an engaging classroom atmosphere. Several other novel capabilities of JavaWIDE such as the history explorer, hyperlinked source code and embedded Java API will also be demonstrated. The presenters will describe the purpose for including each feature within JavaWIDE and will discuss the pedagogical advantages of using these features.

GRIDWORLD & OTHER FRAMEWORKS

JavaWIDE easily integrates with frameworks that are designed to be compatible with unsigned applets, and JavaWIDE can also be used with other frameworks as well, even if they include file input and output or make outside network connections. This part of the workshop will give an overview of how to import and use frameworks within JavaWIDE, and the strategies that are used to adapt applications to fit the security restrictions of unsigned applets.

FILE I/O IN APPLETS

An interesting part of any online IDE is how it handles file input and output and security. Since untrusted applets cannot do file input and output, no online IDE that is applet based can directly perform file input and output. Unsigned applets can perform URL input and make network connections to the server from which they were downloaded. JavaWIDE transforms programs that have file I/O into programs that can run as applets by redirecting file input and output via a safe mechanism that is not a security risk. This redirection happens seamlessly and is very easy and straightforward to use. This part of the session will talk about the capabilities and limitations of JavaWIDE's file I/O and will discuss the relevant design considerations that led to this structure.

STUDENT FEEDBACK, SOCIAL NETWORKING & QUESTIONS

This final part of the workshop will discuss student feedback from those using JavaWIDE in and outside of class over the past two years. The social networking capabilities of JavaWIDE will be demonstrated and the student perception of this social networking integration will be discussed. The workshop will conclude with questions and answers and future plans for JavaWIDE.

USING DATA MINING TO INTRODUCE DATABASES*

NIFTY ASSIGNMENT

Matt Brown
Department of Computer and Information Science
Arkansas Tech University
Russellville, AR 72801
(479) 356-2161
hbrown11@atu.edu

In this CS0 assignment, students collect and clean data, create simple relational databases, and then mine their data for interesting rules. Data mining provides a nifty way to motivate students as they learn basic database concepts. The process of using automated algorithms to discover useful nuggets of previously unknown knowledge is not only intriguing, but also allows insight into the value of storing data.

In the initial stages of the assignment data are generated and cleaned before later insertion into a database. Students fill out surveys about their personal interests and preferences (favorite dessert, TV show, holiday, etc.). The data are consolidated into a spreadsheet and then students must clean the data (correcting misspelled words, eliminating inconsistencies in terminology, etc.). This data quality step demonstrates one of the more important tasks involved in creating modern databases and warehouses. Tables are then created to teach the concepts of relational databases and normalization, requiring students to decompose the tables in a way that reduces storage space but keeps the tables linked for updates and queries. At this point, students run simple SQL queries to answer questions such as "What is the most preferred dessert?"

Finally, students are given an unsupervised association analysis algorithm to mine the database. Students discover affinity rules within their survey answers. Often unexpected results are generated such as {Christmas}® {American Idol}, i.e. students whose favorite holiday is Christmas are nine times more likely to pick American Idol as their favorite television show.

This assignment provides a memorable hands-on experience while demonstrating many important database concepts. Any database supporting standard SQL could be used to implement the assignment. It could be tailored to fit different skill levels—requiring more advance students to normalize the database and/or develop the data mining algorithm on their own. The survey results can also be appended over the course of several semesters to increase the size of the database. The full assignment description and supporting files can be downloaded at http://www.ccsc-ms.org/nifty/10/bro/.

DATA MODELING FOR REAL*

NIFTY ASSIGNMENT

Donna Wright
College of Science, Technology, Engineering & Math
University of Arkansas-Fort Smith
Fort Smith, Ar. 72914
(479) 788-7903
dwright@uafortsmith.edu

INTRODUCTION:

This assignment is suitable for entry level database students in their first data modeling class. It is assigned after students have a good grasp of fundamental database design but does not require an understanding of normalization. The assignment addresses the challenge of helping students relate the dry concepts of database design to their own "real world". When students can visualize the interactions between the data in tables they are designing and real people's lives, they will better understand how and why to apply database design concepts. This assignment can be found at http://www.ccsc-ms.org/nifty/10/wri/

ASSIGNMENT OVERVEW:

This assignment includes two parts - a homework assignment and an in-class activity performed in the following class session. The homework assignment requires students to select a document from their own life and write up proposed entities, attributes and relationships and entity relationship diagram (ERD) that might be relevant to a related database.

The in-class activity puts students in groups to analyze their documents and supporting information and then demonstrate selected documents to the class along with guided classroom discussion

.

LEARNING OUTCOME:

This assignment led to lively student discussion about documents as varied as a restaurant receipt, fishing license, immunization record, and one of the most interesting - a fire report. It helped students recognize that the data they are learning to model is "real" and is a dynamic part of their everyday world. Database design issues were

---

brought into focus as the students recognized problems they have encountered in relation to their document that result from poor database design. Example: The student who was a fireman discussed issues he encounters when trying to file the fire report and students were able to recognize likely design flaws in the database that were causing his issues.

HANDWRITTEN CHARACTER RECOGNITION*

NIFTY ASSIGNMENT

Gabriel J. Ferrer
Department of Mathematics and Computer Science
Hendrix College
Conway, AR 72032
(501) 450-3879
ferrer@hendrix.edu

In this Nifty Assignment, for use in an Artificial Intelligence course, students are given code for a GUI in which they can draw and save handwritten characters on a 20x20 grid. Both the provided code and student solutions are written in Java. They employ a self-organizing map (SOM), which is a type of neural network that reduces a high-dimensional input space into a lower-dimensional output space. It is trained by means of an unsupervised learning algorithm. They must implement the following:

1. The Self-Organizing Map unsupervised learning algorithm, using a threshold neighborhood function.
2. A means of encoding binary drawings as SOM inputs.
3. A classification algorithm that translates SOM outputs into labels.

Once the implementation is complete, students can use the GUI to create, train, and view the SOMs. To view the SOM, students may either:

1. Move the mouse across a grid representing the SOM output nodes; the "ideal" input for the node will appear in the drawing area, or
2. For the drawing currently on display, show the intensity of match for each SOM output node on a color-coded output map.

Once the students have implemented their SOMs and experimented with them a bit, they are tasked to do a series of experiments. In these experiments, they assess the impact of varying the number of SOM iterations, the number of output nodes, the classification algorithm, and the neighborhood function. Once the experiments are complete, the students write a report describing their findings.

This assignment fulfills several course goals:

1. Students apply a machine learning technique to a realistic application domain.

2. As this infrastructure can also be used with a multi-layer perceptron (or any other machine learning technique), our students were able to compare the performance of the SOM and the MLP on a common application domain.
3. Students gain practice with the scientific method.

Nearly all of our students were able to create very effective character recognition learning algorithms using this infrastructure. The full assignment description and supporting files can be downloaded from: "http://www.ccsc-ms.org/nifty/10/fer/".

MAINTAINING CONTROL OF A ROBOT'S LIMBS USING THE BAKERY ALGORITHM *

Patrick McDowell, Theresa Beaubouef
Department of Computer Science and Industrial Technology
Southeastern Louisiana University
Hammond, LA 70402 USA
(985) 549-2189
{pmcdowell, tbeaubouef}@selu.edu

ABSTRACT

This paper discusses a technique for maintaining control of robotic equipment given a single control line for both actuator control and limb position sensing. In particular, the bakery algorithm is used for preventing the problem of command/response packet collisions. These techniques can benefit any robot builder using a servo controller/IO board linked to a command and control computer via single control channel where there is a chance of collisions.

INTRODUCTION

At our university we are currently pursuing a robotics research program focused on providing robots with the ability to adapt and learn as necessary in order to keep their goals in target. That being said, our program is in its early stages, and one of the immediate tasks is to create the infrastructure necessary in order to do the research. This step is a necessary one; valid research cannot be done without working hardware platforms and the low level software to control the actuators and collect the sensor data. To that end we have embarked on a handful projects, one being an automated tennis court cleaning system [1], another being a robotic air hockey opponent [2], and the project that is the subject of this paper: the large actuator test bed.

While there are many novel examples of robotic technology, we are pursuing a path that we hope will result in the development of a series of cost effective "blue collar" robots. That is, for an end result, we want our robots to be able to perform some useful

work. Much as MIT's Big Dog is designed to carry heavy (upwards of 300 lbs) [3] loads across uneven terrain, our long-term goal is create systems that can assist humans in doing work, much as horses and mules have in the past, and tractors do now.

In order to do so, we must be able to control actuators whose force output is in the range of 20 to 1000 pounds of linear thrust, instead of that of typical hobby servos whose peak output is generally from 40 to 350 ounce inches of torque. At first glance it may not seem like that much of a difference, but some quick calculations show that at the low end of the spectrum there is about a 96 to 1 increase in strength, while the high end comes close to a 550 to 1 increase in strength. Given these large differences, in order to lessen equipment breakage, and avoid getting ourselves in a pinch (literally), we are taking a methodical approach to control by building a test bed to facilitate equipment and low-level algorithm development.

This paper details these activities and specifically focuses on our technique of maintaining control of the equipment given a single control line for both actuator control and limb position sensing. While we use a PC linked to an SD 84 servo controller board [4], these techniques can benefit any robot builder who is using a servo controller/IO board linked to a command and control computer via single control channel where there is a chance of command/response packet collisions.

This paper is organized as follows: the background section provides information on typical servo operation versus large actuator control, the approach section describes the bakery algorithm and how its use solved control and sensing conflicts inherent in our application, and finally the conclusions and future work describes issues with the project and the directions in which we plan to head.

BACKGROUND

Many of the robots available in kit form from vendors or those built by researchers and hobbyists use radio control (RC) servos to provide the muscle for moving the joints and ultimately allowing the robot to accomplish a myriad of motions, such as walking, dancing, punching, and arm waving. RC servos were originally used to move the control surfaces and motor throttles of RC planes. The servo itself is a small box (see Figure 1), usually about the size of a small cell phone. On one side of it is an arm, called the horn, which typically can rotate between 0 to about 225 degrees. By attaching a rod from this horn to a control surface, such as a rudder on an RC plane, the servo can move the rudder to the position requested by the plane's pilot.

Figure 1. Typical RC servo. The circular white plate on the top of the box is the horn. The horn has a range of rotational motion from 0 to about 225 degrees. Power, control and ground signals enter the servo through the multi-colored cord as seen in the figure.

The servo's position is proportional to the duty cycle of the signal generated by the pilot's remote control. One of the beauties of the RC servo is that it has an internal system that detects the horn's current location and works through a feedback loop to move the horn to the position indicated by the input control signal. This system works independently of the remote, with the implication being that the remote can send a position signal and let the servo handle the details of moving to the position requested. So essentially the communication from the pilot's controller to the plane is one way.

This method is great, until more force than the strongest RC servo can supply is needed, or one would like to monitor the servo's progress. In the latter case, there are some specialty servos that can provide feedback, but they are expensive, and they have no more torque than the run-of-the-mill servos of the same size.

Consider a situation in which a walking robot gets one of its legs in a bind; how can this be detected? The position information of the actuator is not readily available. A separate sensor at the joint can be used to monitor its progress, but in order to get this information the sensor must be polled. If the sensor is attached to an analog to digital (A2D) port on the servo controller board, the channel from the computer to the controller board will have to be a bi-directional connection, opening the possibility for a collision.

Figure 2. An assortment of linear actuators. Linear actuators work by turning a threaded rod with an electric motor. The threaded rod in turn extends the actuator arm from the actuator's case. In the figure, the arm is the narrow end of the actuator.

Our end goal is to be able control large actuations such as linear actuators or hydraulic cylinders. Unlike RC servos, these systems do not have internal feedback controllers, so they require that the computer monitor their progress by polling a sensor so the actuator can be shut down when it approaches its target position. Given these requirements, and the PC-to-SD84 control system, a reasonable way to manage the communication line between the PC and the controller board is needed. The requirements are as follows:

- *Speed*. The sensors need to be polled on a regular interval, at about 50 times per second—not blazing by any stretch, but still reasonable for getting information from the computer to the controller and reading the response on a serial line.
- *CPU usage*. The system cannot be CPU intense. Controlling the low level actuators is only one of many requirements being asked of the computer.
- *Fairness*. There cannot be a situation in which a sensor response back to the computer, or an actuator stop command, has to wait an indefinite amount of time because the control line is busy.
- *No collisions*. This point cannot be over stressed. If the control board misses an actuator shutdown command because it collides with a sensor position response, the result with even 20-lb strength actuators can be catastrophic. It was these types of events that motivated this work. Out of control linear actuators can flip chairs, move tables, bend metal work carts, and kick researchers with much more starch than ever possible with RC servos.

APPROACH

The basic system consists of a laptop PC linked via a USB cable to an SD84 84 Channel Servo Driver Module made by Devantech. Among this board's features are eighty-four configurable channels, with up to thirty-six being 10-bit resolution A2Ds, or any combination of pulsed width modulation (PWM) and digital I/O. Note that the reference voltage for the A2Ds is 5-volt TTL level, but the channel's voltage signal going to the sensor is at battery level. This means that initially, if the battery has 7.2 volts (the normal amount for a servo battery) the A2D will read maximum value (1024) at every sensor reading at 5 volts or above. The implication is that roughly one third of the sensor range is at maximum value. The solution is to use a regulated 5-volt supply for the A2D channels.

Basic communication with the board is a straightforward programming affair in which a serial communications port is opened and data packets are sent to and fro using the board's command language. For this effort a console program was written in Visual C++ 6.0.

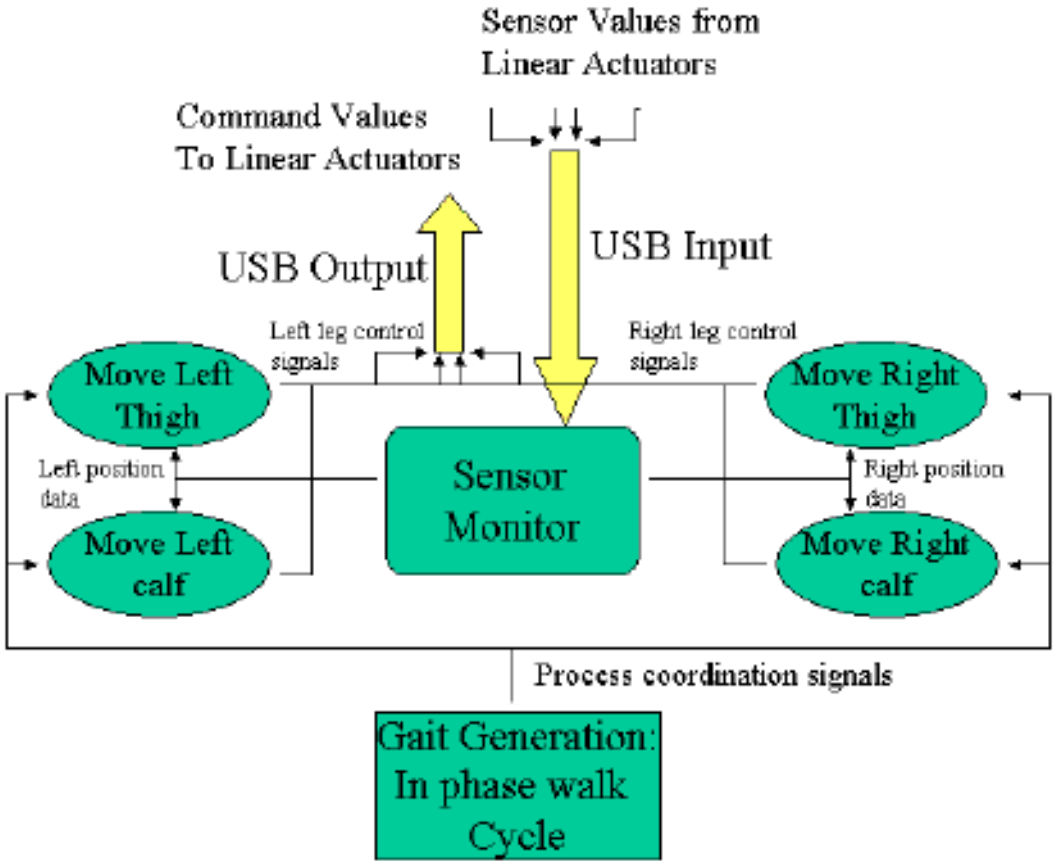


Figure 3. Block-level view of the control software. The gait generator resides in the main program, while the sensor monitor and all leg movement routines run as separate threads.

The control software is set up as follows. The main program starts a sensor monitoring thread whose job is to read the potentiometers in the linear actuators so that

the degree of extension is always known to the leg part (calf, thigh, as seen in Figure 3) controller threads. The sensor monitor thread operates in a continuous loop, alternating between reading the sensors and sleeping. The sleeping is necessary in order to let the other processes have CPU slices.

A linear actuator provides the muscle for each leg part's movement. To control a linear actuator, a PWM port on the SD84 board is connected to a Victor motor controller that in turn provides voltage to the actuator's motor. The Victor controller is basically an amplifier whose input is the same as that of an RC servo (it is a PWM input) and whose output is a voltage between –12 and +12 volts. The leg part controllers (left thigh, right calf etc.) manage movements directed from the gait generation module in the main program. In order to move a part to a particular position, the leg controller sends a signal to the Victor amplifier, which in turn provides the linear actuator with the voltage to get it moving. Then the leg part controller enters a monitor/sleep loop and stops the leg when it reaches its ending position.

All actuator control and sensor reading commands go through the USB line to the SD84 board. Since the parts on a leg are moving in phase with one another, and directly out of phase with their counterparts on other legs, if left unchecked, conflicts between sensor values coming from the control board and actuator commands going from the computer to the control board will occur. In order to assure that no conflicts occur, Lamport's bakery algorithm [5] was used to manage the communications.

The bakery algorithm is usually presented as a tool for managing critical sections in an operating system setting, such as a piece of memory that producer and consumer processes share. The algorithm is extensible to N processes and fulfills the critical section requirements of mutual exclusion (only one process can use the resource at a time), progress (processes that wish to enter a critical section can participate in the decision about which process enters next), and bounded waiting (the wait time is not indefinite and a process can only be deferred a set number of times). The pseudo-code, along with clarifying comments, for a two-process bakery algorithm is shown below.

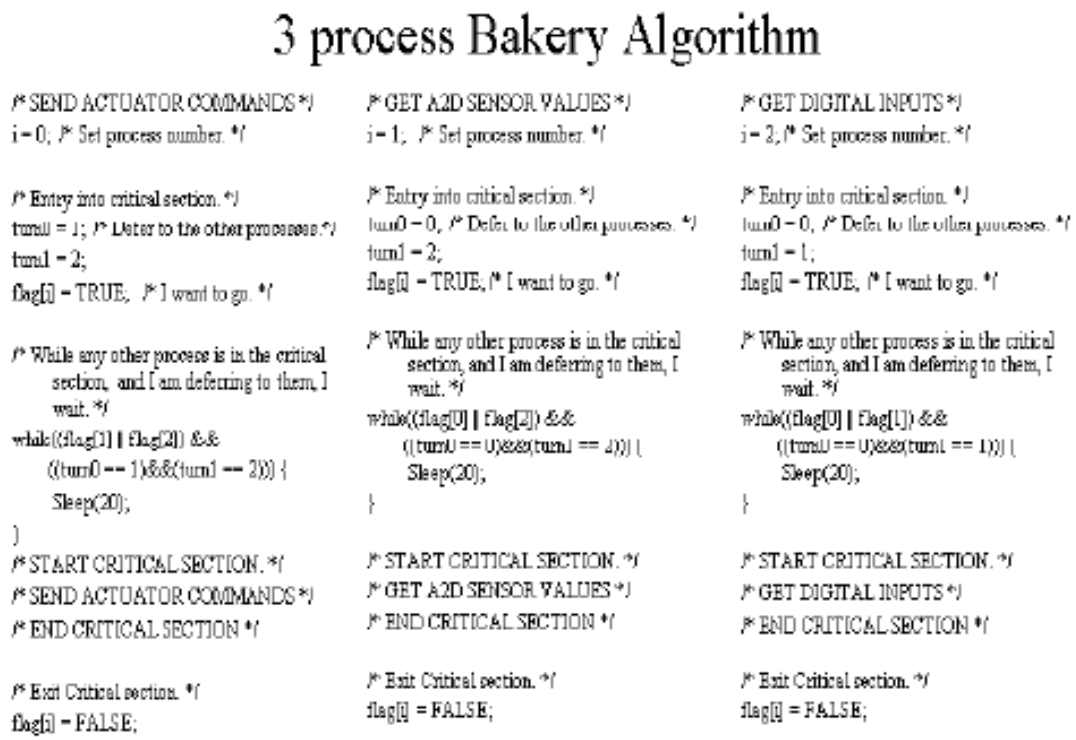
```
 Process P(i)                          /* The variable i = (0, 1) */
Entry: flag[i] = TRUE;                 /* This process wants to enter the critical section. */
Turn = 1-i;                            /* This process defers to the other process. */
While(flag[i-1] && (turn = 1 – i)) { /* Wait while the other process is in the critical
        Do nothing;                        critical section (flag[i – 1] is TRUE) and the
}                                          other process has not deferred to this process. */

/* Here is the body of                 /* In our case, we use the USB serial line to
   code to be executed in                 get a set of sensor values, or send a command. */
   the critical section.
*/                                     /* When complete, we indicate to the other
 Exit: flag[i] = FALSE;                process that we no longer need the critical section,
                                       thus allowing it to enter. */
```

The purpose of the flag variable is to indicate that a process wants to enter a critical section, and that once it is in the critical section the other process cannot enter until it is reset. The Turn variable breaks any ties, that is, if both processes hit the entry point at the same time (they set their flags to TRUE), then (assuming a one CPU

situation) the last process to set the Turn variable will wait for the other process (process 0 sets Turn to 1, process 1 sets Turn to 0, thus allowing process 0 to enter the critical section).

Figure 4 shows our implementation of a 3-process bakery algorithm. The algorithm protects the 3 critical sections (for process numbers i = 0, 1, and 2) in which data is moved between the PC and the servo controller/A2D board.



3 process Bakery Algorithm

```
/* SEND ACTUATOR COMMANDS */
i = 0; /* Set process number. */

/* Entry into critical section. */
turn0 = 1; /* Defer to the other processes. */
turn1 = 2;
flag[i] = TRUE;  /* I want to go. */

/* While any other process is in the critical
      section, and I am deferring to them, I
      wait. */
while((flag[1] || flag[2]) &&
     ((turn0 == 1)&&(turn1 == 2))) {
     Sleep(20);
}
/* START CRITICAL SECTION. */
/* SEND ACTUATOR COMMANDS */
/* END CRITICAL SECTION */

/* Exit Critical section. */
flag[i] = FALSE;
```

```
/* GET A2D SENSOR VALUES */
i = 1;  /* Set process number. */

/* Entry into critical section. */
turn0 = 0, /* Defer to the other processes. */
turn1 = 2;
flag[i] = TRUE, /* I want to go. */

/* While any other process is in the critical
      section, and I am deferring to them, I
      wait. */
while((flag[0] || flag[2]) &&
     ((turn0 == 0)&&(turn1 == 2))) {
     Sleep(20);
}

/* START CRITICAL SECTION. */
/* GET A2D SENSOR VALUES */
/* END CRITICAL SECTION */

/* Exit Critical section. */
flag[i] = FALSE;
```

```
/* GET DIGITAL INPUTS */
i = 2, /* Set process number. */

/* Entry into critical section. */
turn0 = 0,  /* Defer to the other processes. */
turn1 = 1;
flag[i] = TRUE,  /* I want to go. */

/* While any other process is in the critical
      section, and I am deferring to them, I
      wait. */
while((flag[0] || flag[1]) &&
     ((turn0 == 0)&&(turn1 == 1))) {
     Sleep(20);
}

/* START CRITICAL SECTION. */
/* GET DIGITAL INPUTS */
/* END CRITICAL SECTION */

/* Exit Critical section. */
flag[i] = FALSE;
```

Figure 4. This figure shows the bakery algorithm/critical section code for 3 processes in C. The code sections are nearly identical, except for the process numbers and references to the turn0 and turn1 variables.

CONCLUSION

The bakery algorithm provides a quick and easy-to-implement solution for problems that need a fair way to administer mutual exclusion for a group of processes and/or resources. By using this algorithm versus an OS dependent solution such as a semaphore, execution speed is preserved. The "Sleep" command works to prevent busy waiting, resulting in a system whose CPU slice usage is minimal. While the algorithm is usually discussed in reference to operating systems, it has provided an ideal solution for low-level communication management for our robot's hardware/software interface.

ACKNOWLEDGEMENT

This research is supported by Louisiana Board of Regents LEQSF (2007-10)-RD-A-27.

REFERENCES

1. Beaubouef, T., McDowell, P., Ice [Air] Hockey and Tennis Balls: Playing at Computer Science Research with Robotics, *SIGCSE Bulletin*, December 2007

2. Lakin, M., A Robotic Air Hockey Opponent; *CCSC-SC* Hammond La., April 2009

3. Boston Dynamics, "BigDog - The Most Advanced Rough-Terrain Robot on Earth," http://www.bostondynamics.com/robot_bigdog.html, October, 2009.

4, SD84 Technical Specification; http://www.robot-electronics.co.uk/htm/sd84tech.htm

5. Silberchatz and Galvin, *Operating Systems Concepts*, 5th Edition, Addison Wesley, 1998.

A PARETO-OPTIMALITY BASED ROUTING AND WAVELENGTH ASSIGNMENT ALGORITHM FOR WDM NETWORKS*

David L. Sonnier
Lyon College
Batesville, AR 72501
(870) 307-7270

ABSTRACT

A routing and wavelength assignment algorithm is presented for optical communication systems. The algorithm, which is robust, flexible, and computationally feasible, is a hybrid fixed-alternate and adaptive routing algorithm based on Pareto optimality.

INTRODUCTION

The efficient use of optical networks requires algorithms that take advantage of the current technology to achieve maximum use of the optical transmission media. In Wavelength Division Multiplexing (WDM) this is achieved through efficient *routing and wavelength assignment* (RWA) algorithms. This class of algorithm has as its objectives the routing and assignment of wavelengths according to the demand. Let $G = (V, E)$ be a graph where each edge represents an optical fiber link between endpoints. For any pair $p = (s, d)$, a request for a connection from node s to node d is satisfied by assigning a route from among a set of candidate routes, and then assigning a wavelength to carry the information over the selected route.

The problem of satisfying all requests using a minimum number of wavelengths is NP-Hard in general. It can be solved in polynomial time in certain networks. [1] The main objective of this paper is to show that the RWA problem can be formulated as a Multicriteria Shortest Path Problem (MCSP), and then solved using common techniques applicable to the MCSP.

Background

Multiobjective network optimization requirements are occasionally found in network design problems. In such cases, the requirement would typically be to simultaneously optimize competing objectives in a communications or transportation network. Given more than one objective to be simultaneously considered in finding the optimal route from source *s* to destination *d* in a network, our problem becomes the MCSP. For routing and wavelength assignment, let:

$R_0^p, R_{1P}, R_2^p, \ldots, R_n^p$ be candidate lightpaths (routes) for source-destination pair $p = (s, d)$,

W be the number of wavelengths available,

$D(R_j^p)$ be the distance for a particular lightpath (route) for a source-destination (s, d) pair,

$B(R_j^p)$ be a cumulative measure of the number of occupied wavelengths on every link (u, v) from s to d.

We seek routes based on simultaneously minimizing the route lengths and utilizing links which are least utilized. What we are seeking, then, is the *Pareto-optimal* set of solutions. Let R^p be the entire set of candidate routes for pair $p = (s, d)$:

$$R^p = \left\{ R_0^p, R_1^p, R_2^p, \ldots, R_n^p \right\}$$

Assume we have a vector of $K \geq 2$ criteria to be minimized for each route:

$$f_1(R_j^p), f_2(R_j^p), \ldots, f_K(R_j^p)$$

We define $R_{opt}^p \subseteq R^p$ as that subset of candidate routes having the property that, for all routes $R_a^p \in R_{opt}^p$ there is no route $R_b^p \in R^p$ such that

$$f_i(R_b^p) \leq f_i(R_a^p), i = 1..K, \text{ and } f_j(R_b^p) < f_j(R_a^p) \text{ for some } j \in \{1..K\}. \quad (1)$$

In this paper we confine our model to the two-objective case ($K = 2$) since we seek to simultaneously optimize two objective functions: minimize the number of links used in each selected route, and at the same time select a route in which the number of wavelengths being used is minimal (or conversely, the number of available wavelengths is maximal). There may be other considerations such as minimizing the total number of switching operations at a node, or fairness. The proposed algorithm can be applied when $K > 2$, but we shall focus on these two criteria. We make several assumptions for the purpose of demonstrating the algorithm described herein. We assume that the approximate demands on the network are known *a priori*, that the demand on the network is symmetric (i.e., traffic from *s* to *d* will be approximately the same as that from *d* to *s*), and that wavelength conversions along the route are not allowed. This last constraint is known as the wavelength continuity constraint, and may be relaxed in actual implementation.

Project Goals

Based on this view of the problem we can see a multiobjective problem formulation, in which we seek to minimize two competing objective functions, $B(R_j)$ and $D(R_j)$ for all j, subject to a set of routing constraints, topology constraints, and lightpath assignment constraints. Once the problem is reduced to a multiobjective formulation, it may be solved using a number of methods. A multiobjective problem with two objective functions lends itself to several algorithmic solutions, depending on whether the complete solution set or an approximation of that solution set is sought and also depending on whether or not there is a requirement for real-time solution set construction. Regardless of these conditions, the routing and wavelength assignment problem presents an example of a problem that can be solved using multiobjective problem solving and presents an opportunity for research for upper level Computer Science students.

METHODOLOGY

A general solution to the MCSP is described in [2], and summarized along with other methods in [3], where the dimensional intractability of the problem is shown. In practice there is often an upper bound on the number of Pareto-optimal solutions to multicriteria shortest path problems, based on ratio restrictions. [4,5]

In this problem we are seeking to select routes with the shortest distance and the least-used links. Using a weighed sum approach, it is possible to reduce the problem to a single-objective problem and solve it using Dijkstra's algorithm but the solution set is not complete. Consider the cost function defined as follows:

$$C(R_j^p) = \alpha B(R_j^p) + (1-\alpha) D(R_j^p), 0 \le \alpha \le 1 \qquad (2)$$

where $B(R_j^p)$ and $D(R_j^p)$ are as described above. Using a "weight" such as α, a two-objective function can be reduced to a single-objective function. We may obtain a set of solutions by applying various values of α to the objective function (2) and solving the problem multiple times using Dijkstra's algorithm [6,7], but this methodology, while computationally feasible [7], has severe drawbacks. This method reveals a Pareto-optimal set of solutions, each solution being optimal depending on the importance of wavelength utilization or hop-length reduction. Unfortunately, approaching the problem in this manner excludes viable candidate solutions which may not lie on the convex hull of the Pareto-optimal solution set. Another shortcoming of this approach is that we have little knowledge of the desired value of α. A similar formulation is found in [8] (p. 76) but it requires two constants instead of one, and the optimal value must be determined experimentally. This paper describes an approach for reducing the problem of routing and wavelength assignment to the MCSP and then solving the problem. The algorithm described herein determines α dynamically based on network conditions and dynamically maintains a list of routes R_{opt}^p for each pair.

RWA Algorithm

The following algorithm, which is a hybrid between fixed-alternate and adaptive routing [9], provides a set of Pareto-optimal paths between all pairs (*s*, *d*).

1. Solve the All-Pairs Shortest Path Problem to find the most direct route, or $D(R_j^p)$ between node pairs.
2. Apply expected demands to obtain estimated $B(R_j^p)$ for each node pair.
3. Using the two matrices generated in Steps 1 and 2 we solve the problem as a Multi-Criteria Shortest Path problem, generating a Pareto-Optimal set of paths, R_{opt}^p between all pairs *p*.
4. Order R_{opt}^p by increasing $D(R_j^p)$ for all *p*.
5. Compute α.
6. For all pairs (*i*,*j*) use the route that minimizes (2) while keeping remaining paths of R_{opt}^p as alternates.

Step 1 is only executed one time. We reach a balance of simultaneous optimization of *B* and *D* by continuing to adjust the value of α used as we monitor the traffic flow. The desired value of α is determined based on the congestion on the most used link in the network. If the most used link is approaching maximum utilization we would want $0 << \alpha \leq 1$ to reduce the importance of the distance in route selection. If the difference between most-used and least used is minimal we set $0 \leq \alpha << 1$ so that routing decisions are based more on distance than on network conditions. Therefore we let α be the percent utilization of the most used link, (*u*, *v*), in the network:

$$\alpha = \frac{B(R_1^{(u,v)})}{W} \qquad (3)$$

We now have a fixed-alternate scheme for selection of routes, wherein we select candidate routes with decreasing $B(R_j^p)$ and increasing $D(R_j^p)$ from our ordered list depending on the network traffic conditions. Periodically Steps 2-7 of the algorithm are repeated, with "expected demands" replaced by "existing network conditions" in Step 2.

RESULTS

Consider the network in Fig. 1, in which we assume that *W* is uniformly 10 on each link. We determine a set of routes between all pairs using classical routing techniques, then determine the flow over each link of the network based on expected demands. This gives us the occupied wavelength matrix (Fig. 3), the number of occupied wavelengths on each link that would result from simple shortest-path routing. Now we solve the

problem to find R_{opt}^{p} for all pairs *p*. Table 1 shows several Pareto-optimal sets of routes for various pairs (*s*, *d*) based on the expected utilization matrix in Fig. 2.

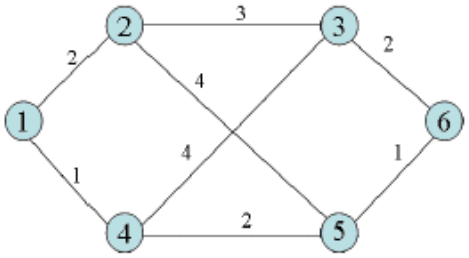


Fig. 1. Sample Network

	1	2	3	4	5	6
1	0	3	1	1	2	1
2	3	0	2	3	0	1
3	1	2	0	1	0	1
4	1	3	1	0	1	1
5	2	0	0	1	0	4
6	1	1	1	1	4	0

Fig. 2. Static Demand Matrix

Based on the congestion of the most-used links, (1, 2) and (1, 4), lightpaths are selected from R_{opt}^{p} initially using α = 0.7, and then alternate routes are selected as necessary.

Distance

	1	2	3	4	5	6
1	0	2	0	1	0	0
2	2	0	3	0	4	0
3	0	3	0	4	0	2
4	1	0	4	0	2	0
5	0	4	0	2	0	1
6	0	0	2	0	1	0

Occupied Wavelength

	1	2	3	4	5	6
1	0	7	0	7	0	0
2	7	0	4	0	0	0
3	0	4	0	1	0	4
4	7	0	1	0	5	0
5	0	0	0	5	0	6
6	0	0	3	0	6	0

Fig. 3. Criteria Matrices

Table 1. Pareto Optimal Routes $(D(R_j^p), B(R_j^p))$

$R_1^{(1,5)}$: 1 – 2 – 5	(6,7)
$R_2^{(1,5)}$: 1 – 4 – 5	(3,12)
$R_1^{(1,6)}$: 1 – 4 – 3 – 6	(7,12)
$R_2^{(1,6)}$: 1 – 4 – 5 – 6	(4,18)
$R_1^{(3,5)}$: 3 – 2 – 5	(7,4)
$R_2^{(3,5)}$: 3 – 4 – 5	(6,6)
$R_3^{(3,5)}$: 3 – 6 – 5	(3,10)

Wavelength assignment is made based on First Fit (FF). Periodically the occupied wavelength matrix is replaced based on current conditions and the set R_{opt}^{p} is recomputed. In the event of failure of one or more links the algorithm is repeated beginning with Step 1.

CONCLUSION

The algorithm described herein proposes a methodology for routing and wavelength assignment by reducing the problem to a multiobjective formulation. In that case the problem being solved is the Multicriteria Shortest Path Problem (MCSP). Although the MCSP is NP-Hard in general, computational studies have shown that it is rare to find real-world problems in which the size of the solution set and the computational time required to find it are exponential. This is not necessarily true for other classes of multiobjective problems, such as the Multicriteria Spanning Tree problem, for example, in which both the solution set and the time to find it balloon exponentially for even a small or sparse network.

To accommodate fluctuating demands over the network, the algorithm must include periodic evaluation and recalculation of the route assignments. Further analysis and study of this algorithm could provide senior level Computer Science students with an opportunity for a computational study. The next level of study would involve simulation using real data, and possibly a hardware implementation. However, even as a mere theoretical solution to the routing and wavelength assignment problem, this methodology presents an example of a situation in which a problem can be cast as a multiobjective problem and then solved using multiobjective problem solving techniques.

REFERENCES

[1] R. Dutta and G. N. Rouskas, " Traffic Grooming in WDM Networks: Past and Future", IEEE Network, November/December 2002, p. 46-56.

[2] E.Q.V. Martins, "On a Multicriteria Shortest Path Problem," *European Journal of Operational Research*, **16**, 236-245 (1983).

[3] M. Ehrgott, *Multicriteria Optimization*, Lecture Notes in Economics and Mathematical Systems (Springer-Verlag Berlin/Heidelberg, 2000).

[4] M. Müller-Hannemann and K. Weihe, *Pareto shortest paths is often feasible in practice*, Lecture Notes in Computer Science, pp. 185-197 (Springer., 2001).

[5] M. Müller-Hannemann and K. Weihe, *On the cardinality of the Pareto Set in Bicriteria Shortest Path Problems*, Annals of Operations Research, vol 147, pp. 269-286, Springer. (2006).

[6] J. Mote, I. Murthy, and D.L. Olson, "A Parametric Approach to Solving Bicriterion Shortest Path Problems," *European Journal of Operational Research,* **53**, 81-92. (1991)

[7] D. Sonnier, "Multicriteria Optimization – Some Parallel Approaches" Presented at WMSCI 2005. *Proceedings, Volume III, The 9th World Multi-Conference on Systemics, Cybernetics, and Informatics*, 30-35. (2003).

[8] Murthy, C. Siva Ram, and Gurusamy, Mohan. *WDM Optical Networks: Concepts, Design, and Algorithms.* Prentice-Hall, 2002. *(this one is not actually referenced?)*

[9] Zhu, Keyao; Hzu, Hongyue, and Mukherjee, Biswanath. *Traffic Grooming in Optical WDM Mesh Networks.* Springer, 2005.

CRITERIA-BASED PARALLELISM FOR MULTIOBJECTIVE PROBLEM SOLVING*

David L. Sonnier, Matt Bradley
Lyon College
Batesville, AR 72501
(870) 307-7270
david.sonnier@lyon.edu

ABSTRACT

Computer scientists typically solve problems that require the optimization of some objective function. Given more than one objective to be simultaneously optimized, the problem becomes a Multicriteria Optimization Problem. Whereas the single objective version of a problem may be solved by use of well researched and studied algorithms, the Multicriteria version of the problem is typically not so quickly and easily solved. In fact, typically there is no polynomial-time algorithm for solving the multicriteria version, and the solution is not a single solution but rather a Pareto-Optimal set of solutions. In some cases it is possible to find an approximation of the solution set by solving *Reduced Criteria Subproblems*, then merging the solutions. This paper provides an overview of this approach, presents a parallel computing solution based on merging reduced-criteria subproblems, and gives consideration to the limitations of this methodology.

INTRODUCTION

A problem requiring that more than one objective function be simultaneously optimized is referred to as a multicriteria or multiobjective problem. Using a weighed sum approach, it is possible to reduce the problem to a single-objective problem and solve it using classical algorithm designed for the single-objective version. Computational studies are described in [1] and [2] with the latter being a parallel implementation. In a multicriteria optimization problem, a supported solution is defined as an extreme point of the convex hull formed in the outcome space. The supported solutions can be found

quickly in parallel using a weighted-sum approach. However only supported solutions are found in this manner. We compare the results obtained in this manner against results obtained using reduced criteria subproblems.

Background

An optimization problem can be generally described as:

$$\text{Min } \mathbf{f(x)} = [f_1(x), f_2(x), ..., f_K(x)], \mathbf{x} \in \mathbf{X}, \quad (1)$$

s.t. constraints

The decision variables are

$$\mathbf{x} = [x_1, x_2, ... , x_n] \quad (2)$$

in decision space ***X***. We say that solution $\mathbf{x}^*$ is Pareto-optimal if there is no $\mathbf{x} \in \mathbf{X}$ such that:

$$\boldsymbol{f_i(x) \leq f_i(x^*), i = 1..K, \text{ and } f_j(x) < f_j(x^*) \text{ for some } j \in \{1..K\}}. \quad (3)$$

An example would be the multiobjective "best" path problem in a transportation network, in which we optimize distance, f_1, but also road quality, f_2, and perhaps two other objective functions, f_3 and f_4 based on other criteria such as maintenance / gas station availability or availability of some other logistics support. The set of "best" paths would be those which make up the Pareto-optimal set, and the decision maker would choose from the set based on the relative importance of the objective functions. The decision variables correspond to the route selected. Unlike the single-objective counterpart, which can be easily solved using Dijkstra's algorithm, finding the complete solution set has been shown to be non-polynomial. [3]

Solutions fall into the category of supported and unsupported. Supported solutions can be found using a set Λ of weights, with the following property:

$$\Lambda = \left\{ \lambda \in R^K, \sum_{i=1}^{K} \lambda_i = 1, \lambda_i > 0 \right\} \quad (4)$$

Using $(\lambda_1, \lambda_2, ..., \lambda_K)$ we reduce the problem defined in (1) to a single-objective problem and solve it using classical algorithms. In essence, then, the problem becomes the single objective one:

$$\text{Min } \lambda_1 f_1(\mathbf{x}) + \lambda_{2f2}(\mathbf{x}) + ... + \lambda_K f_K(\mathbf{x}) : \mathbf{x} \in \mathbf{X} \quad (5)$$

for some set $(\lambda_1, \lambda_2, ..., \lambda_K)$ of weights as defined in (4) above. This provides an easy and convenient approach for providing a reasonable approximation of the solution set in polynomial time. Supported solutions lie on the convex hull of the feasible region of the objective space. For $K = 2$, for example, we can find a set of supported solutions as follows. Define a weight λ such that $0 \leq \lambda \leq 1$ and define the following objective function:

$$\lambda f_1 + (1-\lambda) f_2 \quad (6)$$

A study using this approach can be found in [2]. A parallel method based on the parametric approach is used, with each processor finding the supported solutions residing within a portion of the range of values of λ.

Project goals

We can find a set of supported solutions quickly by generating a series of weighted-sum vectors. This approach leaves out unsupported solutions. We show herein that in some cases it is possible to find a more complete set of solutions in reduced time by simultaneously solving sub-problems involving *K*-1 criteria and then merging the solution sets, as described in [4]. We describe a parallel implementation herein, based on the Multiobjective Shortest Path Problem.

METHODOLOGY

The *Merge* of any two nondominated sets *A* and *B* is defined as the set of *nondominated* vectors in the union of sets *A* and *B*: [5]

$$Merge(A,B) = \{A \cup B\} - \{\mathbf{x} \in [A \cup B | f_k(\mathbf{x}^*) \leq f_k(\mathbf{x}) \text{ for some } \mathbf{x}^* \neq \mathbf{x},\ k \in K,\ \mathbf{x}^* \in A \cup B]\} \quad (7)$$

A *K-1 criteria sub-problem* of a *K*-criteria problem is defined as a subproblem obtained by considering only *K*-1 of the *K* criteria of the original problem. In [4] we find the definition of MOP(*i*) as the following *K*-1-criteria subproblem, with $f_i(\mathbf{x})$ dropped from the set of *K* criteria:

$$\begin{aligned} &\min \mathbf{f(x)} = [f_1(x), \ldots f_{i-1}(x),\ f_{i+1}(x), \ldots, f_K(x)] \\ &s.t.\ x \in X \end{aligned} \quad (8)$$

The solution set resulting from the *Merge* (·) of all *K*-1-criteria subproblems is referred to in [4] as Opt^{K-1}. To obtain a solution by use of the *Merge* (·) of solution sets of reduced criteria subproblems, it is necessary to understand the nature of the subproblems. First we consider the following three limitations of *K*-1-criteria subproblems by illustrating with $K = 3$:

- PROPERTY 1: *Solutions to K-1-criteria subproblems may not be "K-efficient," or Pareto Optimal for the K-criteria problem*. Consider a tri-objective problem with the solution set in Figure 1. Here we can see that even though (3,3,5) is 2-efficient for criteria 1 and 2, it is dominated in the 3-criteria case, by (3,3,1). Therefore the solutions must be filtered, as suggested by Equation (7).
- PROPERTY 2: *Pareto-optimal solutions for the K-criteria problem may not be Pareto-optimal for the K-1-criteria subproblems*. Consider a tri-objective problem in which the solution set is as depicted in Figure 2. We cannot determine all solutions by merging the bi-criteria subproblems. The solution (3,3,3) is Pareto-optimal (in fact it is supported) yet it is not a solution to any of the bi-criteria subproblems; only {(-,2,2), (2,-,2), (2,2,-)}.
- PROPERTY 3: *Supported solutions to K-1-criteria subproblems are not necessarily supported for K criteria*. Consider the problem in which the solution set is as depicted in Figure 3. According to Definition 1, note that the solution

(3,3,5) is supported for criteria 1 and 2, for $\lambda_1 = \lambda_2$. However, it is not supported for the 3-criteria problem for any combination of $[\lambda_1, \lambda_2, \lambda_3]$.

$\left\{\begin{pmatrix}2\\4\\0\end{pmatrix},\begin{pmatrix}3\\3\\1\end{pmatrix},\begin{pmatrix}3\\3\\5\end{pmatrix},\begin{pmatrix}4\\2\\0\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}6\\2\\2\end{pmatrix},\begin{pmatrix}2\\6\\2\end{pmatrix},\begin{pmatrix}2\\2\\6\end{pmatrix},\begin{pmatrix}3\\3\\3\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}2\\4\\0\end{pmatrix},\begin{pmatrix}3\\3\\5\end{pmatrix},\begin{pmatrix}4\\2\\0\end{pmatrix}\right\}$
Figure 1	Figure 2	Figure 3

Assume that a solution is *K*-efficient but not *K*-1-efficient. Then it is dominated in all *K* - 1 criteria subproblems, for any subset of *K*-1 objectives. It may be either a supported solution or an unsupported solution. Consider the nondominated set in Figure 4.

$\left\{\begin{pmatrix}6\\2\\2\end{pmatrix},\begin{pmatrix}2\\6\\2\end{pmatrix},\begin{pmatrix}2\\2\\6\end{pmatrix},\begin{pmatrix}3\\3\\3\end{pmatrix},\begin{pmatrix}2\\4\\4\end{pmatrix},\begin{pmatrix}4\\2\\4\end{pmatrix},\begin{pmatrix}4\\4\\2\end{pmatrix}\right\}$	$\left\{\begin{pmatrix}6\\2\\2\\3\end{pmatrix},\begin{pmatrix}2\\6\\2\\8\end{pmatrix},\begin{pmatrix}2\\2\\6\\4\end{pmatrix},\begin{pmatrix}3\\3\\3\\5\end{pmatrix},\begin{pmatrix}2\\4\\4\\3\end{pmatrix},\begin{pmatrix}4\\2\\4\\6\end{pmatrix},\begin{pmatrix}4\\4\\2\\5\end{pmatrix}\right\}$
Figure 4	Figure 5

Solutions 4-7 are not dominated in *K* objectives, but they are dominated in all *K*-1 objective subproblems. Approximating this solution set using *K*-1 criteria subproblems would reveal only the first three of the seven solutions. On the other hand, consider a solution set with four criteria (Figure 5). Note that the first three objectives are the same as in Figure 4. Approximating this solution set using *K*-1 criteria subproblems would reveal all of the seven solutions. The reason for this is straight-forward. In the pair-wise test for optimality between solution **x** and solution **y**, x may be spuriously eliminated from consideration if it is dominated in all *K*-1-criteria subproblems.

Table I. Possible Outcomes in 2-Criteria Subproblem

k_1	k_2	Possible Outcome
$\mathbf{x}_1$	$\mathbf{x}_2$	**y** Dominated by **x**
$\mathbf{x}_1$	$\bar{\mathbf{x}}_2$	**x** Not Dominated by **y**
$\bar{\mathbf{x}}_1$	$\mathbf{x}_2$	**x** Not Dominated by **y**
$\bar{\mathbf{x}}_1$	$\bar{\mathbf{x}}_2$	**x** Dominated by **y**

Table II. Possible Outcomes in 3-Criteria Subproblem

k_1	k_2	k_3	Possible Outcome
$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	**y** Dominated by **x**
$\mathbf{x}_1$	$\mathbf{x}_2$	$\bar{\mathbf{x}}_3$	**x** Not Dominated by **y**
$\mathbf{x}_1$	$\bar{\mathbf{x}}_2$	$\mathbf{x}_3$	**x** Not Dominated by **y**
$\mathbf{x}_1$	$\bar{\mathbf{x}}_2$	$\bar{\mathbf{x}}_3$	**x** Not Dominated by **y**
$\bar{\mathbf{x}}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	**x** Not Dominated by **y**
$\bar{\mathbf{x}}_1$	$\mathbf{x}_2$	$\bar{\mathbf{x}}_3$	**x** Not Dominated by **y**
$\bar{\mathbf{x}}_1$	$\bar{\mathbf{x}}_2$	$\mathbf{x}_3$	**x** Not Dominated by **y**
$\bar{\mathbf{x}}_1$	$\bar{\mathbf{x}}_2$	$\bar{\mathbf{x}}_3$	**x** Dominated by **y**

To quantify this, let $K = 3$ and consider a solution, **x**, to the *K*-criteria problem, that is under consideration in the 2-criteria subproblem MOP(3). Denote by x_i that solution

x is not dominated by solution **y** in objective function *i*, and denote by $\bar{x}_i$ that solution **x** is dominated by **y** in that same objective function. Then we have the four possible outcomes in the optimality check used in MOP(3) (see Table 1). Assuming uniform distribution of the solutions, there will be a probability of 0.5 that neither solution will spuriously dominate the other, a probability of 0.25 that x will spuriously dominate y, and a probability of 0.25 that y will spuriously dominate x. Therefore, the probability of solution x not being dominated by solution y in MOP(*i*) is 0.75.

For $K = 4$ the optimality check for the *K*-1-criteria subproblems involves 3 criteria. Here again we concern ourselves with those Pareto optimal solutions which are, in reality, 4-efficient being dominated spuriously in a 3-criteria subproblem.

The probability of one solution spuriously dominating another solution in a pairwise comparison is 0.25, and the probability of solution **x** not being dominated in MOP(*i*) is 87.5. And in general, for a *K*-criteria problem solved by decomposition into *K*-1-criteria subproblems, one Pareto-efficient solution will dominate another in a subproblem with a probability of

$$\frac{1}{2^{K-1}} \tag{9}$$

Recalling that each of these comparisons in which one efficient solution dominates another in a subproblem, it follows that for larger *K* we can expect fewer occurrences. Similarly, for 2-criteria comparisons, 50% of the comparisons result in no one solution dominating another; for 3-criteria comparisons 75% result in no one solution dominating another, and for four criteria comparisons 87.5% result similarly, and in general for *K*-criteria comparisons,

$$\frac{2^K - 2}{2^K} \cdot (100) \tag{10}$$

yields the percent of comparisons in which no one solution dominating another. This ratio approaches 100% with increasing *K*. Whether or not it is beneficial to further decompose the subproblems into bi-criteria subproblems depends on the desired completeness of the solution set. The merge of bi-criteria subproblems was one of the methods tested in this study.

RESULTS

To compare various methods, the 4-criteria Shortest Path Problem is used. The graphs used for this study were derived from problem instances taken from the web page of the Konrad Zuse Institute in Berlin [6]. For each problem, one of the sets of edge costs was generated using NETGEN, and the other sets were generated randomly. We compare the results of both time and solution set completeness using four methods: finding the complete solution set, finding the solution obtained using a weighted sum method, finding the solution set obtained by merging bi-criteria subproblems, and finding the solution obtained by merging *K*-1-Criteria subproblems. In each case the problem is solved with an eight-computer cluster using Message Passing Interface (MPI) extensions to the C programming language. In Table III we can see the number of solutions obtained using K-1-Criteria subproblems confirms the predictions made in Equations 9 and 10. Figure

6 and Figure 7 show the computational time and the number of solutions obtained. Note that in Figure 6 the time for generating the complete solution is normally several times greater than the time required to generate the solution by merging *K*-1-criteria subproblems, but in Figure 7 the number of solutions generated using these two methods is similar. For problems of size 500 or less there is very little benefit in using cluster computing, For a problem of size 500 or larger we can see definite benefit in terms of computational time.

CONCLUSIONS

Further studies of these methodologies provide excellent opportunities for senior level Computer Science students. The problems are relatively easy to understand and student contributions can be meaningful. Other areas to investigate are weight set generation, various categories of multiobjective problems, and real world applications of multiobjective problem solving.

Table III. Number of Solutions Obtained

n	Complete	Weighted	Bi-Criteria Sub	K-1 Sub
200	91	25	51	84
300	51	18	33	46
400	42	7	28	40
600	98	19	49	94
800	183	23	57	148
1000	287	42	89	244
1500	56	19	28	53
2000	40	12	33	40

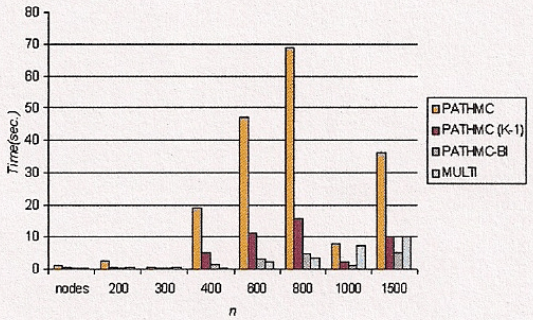


Fig. 6. $K = 4$, Time

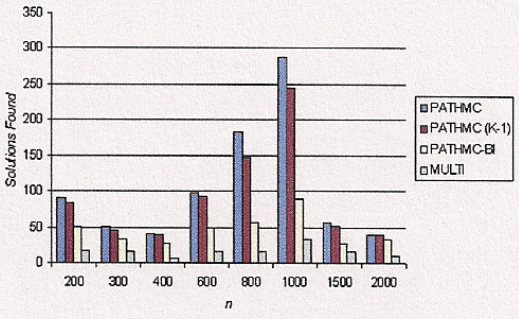


Fig. 7. $K = 4$, Number of Solutions

REFERENCES

[1] HU, J. AND CHAN, Y., A Multi-criteria routing Model for Incident Management. Presented at the 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii October 2005.

[2] SONNIER, D., Multicriteria Optimization - Some Parallel Approaches Presented at WMSCI 2005. *Proceedings, Volume III, The 9th World Multi-Conference on Systemics, Cybernetics, and Informatics*, 30-35, 2003.

[3] Ehrgott, Matthias , "Lecture Notes in Economics and Mathematical Systems: Multicriteria Optimization." Springer-Verlag Berlin/Heidelberg, 2000.

[4] EHRGOTT, M. AND TENFELDE-PODEHL, D., Computation of Ideal and Nadir Values and Implications for their Use in MCDM Methods, *European Journal of Operational Research*,151,119-139, 2002.

[5] BRUMBAUGH-SMITH, J. AND SHIER, D., An Empirical Investigation of Some Bicriterion Shortest Path Algorithms. *European Journal of Operational Research*,43, 2, 216-244, 1989.

[6] ZUSE INSTITUTE, BERLIN. See Minimum Cost Network Flow and Transportation, NETGEN INSTANCES Files, containing instances of the minimum cost flow problem generated by NETGEN or just the NETGEN parameters. www.zib.de/ Service/index.en.html. For generation see D. Klingman, A. Napier, and J. Stutz: *NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost low networks*, Management Science 20 (1974), pp. 814-820, non-standard problems contributed by A. Löbel.

TEACHING A GAME PROGRAMMING CLASS FOR THE FIRST TIME*

TUTORIAL PRESENTATION

Frank McCown
Computer Science Department
Harding University
Searcy, AR 71239
501 279-4826
fmccown@harding.edu

In an effort to increase interest in Computer Science (CS) and offer compelling and attractive electives, many CS departments are contemplating offering a game programming course for the first time. A major obstacle for offering such a course is finding faculty who are already experienced game programmers. Inexperienced faculty may be interested in teaching game programming but may hesitate tackling a subject that students are perceived to be more knowledgeable about; after all, college students are much more experienced playing the latest games and have logged considerably more time on a variety of gaming platforms than most CS faculty.

This tutorial is designed for CS instructors with no background in game programming who are interested in learning how to teach a game programming course for the first time. We will share our experience developing an upper level game programming class at Harding University that uses the XNA platform, a C# programming environment for creating games on the Xbox, Windows, and Zune. We developed our course to actively engage students in learning and teaching others various game programming concepts. Students performed independent research on topics like animation, sound, collision detection, 3D modeling, and deployment, and they shared their findings in a number of presentations given to their fellow classmates. The structure of the course allowed the instructor to gain experience along with the students in learning rather than being pressured to play the "sage on the stage." This lessoned the amount of preparation required by the instructor at every class meeting.

In this tutorial, we will introduce the XNA platform and have the participants build a simple Pong game. We will share the various team projects and homework assignments that made our class a success and discuss lessons learned from teaching this course the first time.

PRESENTER BACKGROUND

Dr. Frank McCown is an assistant professor of Computer Science at Harding University. Since 1997 he has taught numerous CS courses including Computer Graphics, Graphical User Interface Programming, Internet Development, Search Engine Development, and most recently, Game Programming. His research interests are in search engines, web crawling, and web archiving, but he has had a life-long interest in computer games. Although Dr. McCown had little experience in game programming, he endeavored to teach a game programming class after many students requested the department offer such a course. The class was held in the Fall of 2009 and was well received.

ENCODING ROBOTIC SENSOR STATES FOR Q-LEARNING USING THE SELF-ORGANIZING MAP*

Gabriel J. Ferrer
Department of Mathematics and Computer Science
Hendrix College
Conway, AR 72032
(501) 450-3879
ferrer@hendrix.edu

ABSTRACT

The self-organizing map (SOM) [1] reduces a large input space to a fixed-size output space. It is trained by means of an unsupervised learning algorithm. One application of the SOM is to transform a set of robot sensor readings into a state space suitable as the input for the reinforcement learning algorithm Q-learning [5].

We have implemented two different formulations of this concept [2][3] using the Lego Mindstorms NXT robot [8], a robot commonly used in undergraduate computer science courses. We compared the performance of our implementations against a traditional Q-learning implementation. We found that the number and type of sensors encoded by the SOM has a significant impact on the quality of the behavior the robot learns.

INTRODUCTION

The Q-learning algorithm [5] is a very popular algorithm for reinforcement learning. It is straightforward to implement and tends to learn effective behaviors. Mahadevan and Connell [6] pioneered the use of Q-learning for a mobile robot to learn behaviors in the physical world.

The main drawback of Q-learning from the point of view of robotic implementation is that it requires all possible combinations of sensor values to be reduced to a finite number of states. Furthermore, the rate at which the algorithm converges to a behavior

depends directly on the number of states selected for encoding the sensor values. This becomes particularly troublesome when high-resolution sensors such as sonars are employed.

In order to address this, Touzet [3] and later Smith [2] combined the Q-learning algorithm with Kohonen's Self-Organizing Map (SOM) [1]. The SOM is a type of neural network that learns how to classify its inputs into a fixed number of output nodes. By feeding sensor input into a SOM, the strongest SOM output can be used to define the current state for the Q-learning algorithm.

Touzet [3] implemented and tested his formulation using the Khepera robot [7]. The Khepera is a cylindrical robot of radius 6 cm. It has eight infrared proximity sensors that report distances ranging from 2 to 5 cm. The robots from Smith's paper [2] exist entirely in simulation.

The Lego Mindstorms NXT robot [8] has proven to be popular among undergraduate computer science educators. However, its sensor inputs and performance profile differ significantly from the robot platforms used in the previously mentioned papers. Consequently, we decided to implement both approaches using the Lego Mindstorms NXT to assess their utility with this popular platform. We have found that we can reproduce their successful results as long as the state of the motors is not part of the state encoding given to the SOM.

The first two sections describe the Q-learning algorithm and the unsupervised learning algorithms used for training a self-organizing map. The third section describes our experimental setup. We then describe and analyze our results, followed by our conclusions.

Q-LEARNING

In reinforcement learning, the agent learns a behavior based on its expected reward. In the Q-learning implementation of reinforcement learning, expected rewards are stored in a two-dimensional array. One index of the array is the current state; the other index denotes an action to be performed in that state. The value of a particular state/action pair is the expected reward for performing the given action from the given state. The elements of this array are called “Q-values”.

At each time step, the learning agent performs the following tasks:

1. Calculate the state index based on current sensor values.
2. Calculate the reward earned by the most recent action.
3. Update the Q-value for the previous state/action pair.
4. Select and perform a new action based on the Q-values.

The first two steps (calculating the state and reward) are implemented based on domain constraints. The Q-value is updated according to the following formula:

$$Q(s,a) = (1 - \alpha)\, Q(s,a) + \alpha\, (r + \gamma \max(Q(s',a)))$$

In this formula, the term α is called the “learning rate” and γ is called the “discount”. The learning rate controls the speed with which the Q-values change; the discount represents a trade-off between current and future rewards. The state s' is the state that

results from applying action *a* from state *s*, and *r* is the reward received after applying *a* in *s*.

Actions are selected based on the Q-values. From the current state, the action that is predicted to yield the largest reward (i.e. has the largest Q-value) is selected and performed. Formulated in this way, assuming that every state/action pair is visited an infinite number of times, Q-learning will converge to an optimal controller.

Since infinite state exploration is impossible in practice, action selection becomes a trade-off between *exploration* of the state/action reward space and *exploitation* of the predicted rewards stored in the Q-table. A common solution to this problem is to use an additional constant ε. At each time step, if a randomly generated value is greater than ε, the agent will use the best available action. If not, an action will be selected randomly.

SELF-ORGANIZING MAPS

The self-organizing map (SOM) [1] is a type of artificial neural network. Each output node is associated with a vector representing the "ideal input" for that node. The output of a SOM is determined competitively when an input vector is presented to it. Let the *distance* between an input vector and the ideal input for an output node be defined as the square root of the sum-of-squared-differences between the two vectors. We then say that the activated output node is the output node whose ideal input has the smallest distance to the input vector.

The output nodes themselves are arranged in a Cartesian grid. On each iteration, the ideal input for the winning output node as well as some of its neighbors in the grid is modified according to the following formula for each affected output node *i* and ideal input element *j*:

$$weight_{ij} = weight_{ij} + (learningRate * (input_{ij} - weight_{ij}))$$

One approach for determining affected nodes is to set a radius parameter. All output nodes within the radius are updated according to this rule. Another common approach is to use a Gaussian neighborhood function $e^{-d2 / (2c2)}$, where *c* is the radius parameter and *d* is the Cartesian distance in the grid between the winning output node and the output node being modified. The result of the Gaussian is multiplied by the learning rate in the above computation for every output node in the SOM. This approach is used in Smith's implementation [2].

In Touzet's implementation [3], the winning output node uses a constant learning rate of 0.9, and each of its Manhattan neighbors in the Cartesian output grid is updated with a learning rate of 0.4. No other output nodes are updated on a given iteration.

The SOM is biologically inspired by the cerebral cortex. The use of neighborhoods enables the SOM to imitate the cortical phenomenon of regions that specialize in recognizing certain sensory data.

EXPERIMENTAL SETUP

We implemented the combination of Q-learning with a SOM in the Java language using the LeJOS implementation version 0.85 [9] for the Lego Mindstorms NXT robot.

In our implementations, the winning SOM output node is the state index for the Q-table. Each experiment ran for 240 seconds, which in all cases represented 800 iterations of the Q-learning algorithm. All experiments had 36 state indices for the Q-table and three actions: Both motors forward, left motor backward with right motor stopped, and right motor backward with left motor stopped. Our robot was configured with a single forward-facing ultrasonic sensor and two forward-facing bump sensors, one on the front left, the other on the front right.

We used two different Q-learning implementations without a SOM as controls. In the first implementation (designated **Qa**), the state was derived from the combination of each bumper state, the current selected action (out of three), and one of three discretized sonar states. The first discretized sonar state represented distances from 0-19 cm; the second state was 20-39 cm; and the third state was 40 cm or more.

In the second implementation (designated **Qb**), we did not represent the current action at all; instead, we discretized nine sonar states. The first state represented 0-11 cm; the second state 12-23 cm; and so forth up to the eighth state at 84-95 cm. The ninth state was 96 cm or more.

In all cases, the reward was calculated as follows. If either bump sensor is pressed, the reward is 0.0. If neither bump sensor is pressed, if both motors are going forward, the base reward is 1.0; otherwise, it is 0.5. If the sonar distance is greater than 20 cm, the base reward is used; otherwise, it is scaled down depending on how close the sonar value is to zero cm. This reward function is designed to teach the robot to drive forward while avoiding obstacles in its path.

We used four different Q-learning implementations with a SOM. We followed each of Smith [2] (designated **QSOM**) and Touzet [3] (designated **QT**), and for each of these implementations, we implemented both with (**a**) and without (**b**) encoding the motor speeds as part of the state. In all four implementations, both the sonar and each bump sensor were part of the input to the SOM. Consequently, **QSOMa** and **QTa** had length-5 input vectors, while **QSOMb** and **QTb** had length-3 input vectors for each SOM. As with our control implementations, the Q-learning tables each had 36 input states; hence, each SOM had 36 output nodes.

When creating the input vectors from the sensor values, we normalized their ranges as follows. Let the scale factor be 100 / (maxSensorValue – minSensorValue). We multiply the scale factor by the sensor reading to produce a scaled input to the SOM. As the sonar range is 0 to 255, while the motor speeds range from -900 to 900, the intention is to normalize the inputs so that the raw sensor range does not distort their impact on the SOM.

For all experiments, the discount (γ) term for updating the Q-table values was 0.5. For **Qa**, **Qb**, **QSOMa**, and **QSOMb**, the learning rate is calculated using the formula $1/(1 + (t/100))$, where t is the current iteration of the algorithm. The neighborhood radius is the learning rate multiplied by 3. For **QTa** and **QTb**, the learning rate is fixed at 0.9. In both cases, the learning rate for Q-learning and the SOM is identical.

The exploration/exploitation trade-off is handled by setting an epsilon value of the learning rate divided by four. If a randomly generated value between 0 and 1 is below

epsilon, the next action is selected randomly, with a distribution proportionate to the expected rewards. Otherwise, the action with the highest expected reward is selected.

At startup, the ideal input vectors for each SOM are all set to zero, and all of the Q-values are initialized to 0.5.

RESULTS

We ran at least three experiments for each of the six implementations. We ran seven experiments for **Qa** and four experiments for **QSOMa**. We employ the total reward earned over 800 iterations as our primary quantitative performance metric. Our results are in the table below.

	Qa	**Qb**	**QSOMa**	**QSOMb**	**QTa**	**QTb**
Mean	607.97	578.91	468.86	534.49	456.19	545.61
Std. Dev.	81.92	76.95	39.39	160.41	85.07	57.98
Median	608.75	667.5	485.11	587.64	442.62	560.77
Minimum	506.47	528.67	410.2	354.25	378.72	481.55
Maximum	723	540.55	495	661.59	547.22	594.5
Mean/Iter.	0.76	0.72	0.59	0.67	0.57	0.68
StdDv/Iter.	0.1	0.1	0.05	0.2	0.11	0.07

In terms of the quantitative data, the SOM-based solutions perform somewhat worse than the traditional Q-learning implementations. The creation of the "b" implementations was inspired by the particularly poor performance of **QSOMa** and **QTa** relative to **Qa**. When we examined the ideal inputs of a SOM created by a run of **QSOMa**, we observed that the encoded motor speeds ranged from 2% to 50%, while the encoded sonar value was stuck between 90% and 94%. In other words, the SOM learned a state encoding for the sonar that paid no heed whatsoever to any significant variance in sonar input. In the case of **QTa**, a much larger range of sonar values (10% to 100%) is present in the ideal inputs; nevertheless, performance was about equally bad.

We realized that for this task it was not necessary to encode the motor speeds as part of the state. Since the expected reward is conditioned both on the current state and the selected action, it seemed that including the action in the state was redundant. While this change made no significant difference to the performance of our control implementation, it did have a very positive effect on the performance of our SOM-based implementations. In fact, two of the three experiments with **QSOMb** produced obstacle-avoiding behaviors that were arguably superior to those of our control implementations.

For these two experiments, an inspection of the ideal inputs for the SOM is helpful in understanding their behavior. In both cases, the ideal inputs assumed that the bumpers were not pressed at all; for those state elements, there was no variation. However, the sonar value varied from approximately 40% to 95% in both cases. When the sonar readings were at the low end of this range, it was clear from observing the robot's behavior that the high-reward action was to turn. Otherwise, it drove forward. Once the learning algorithm had stabilized, these experiments produced extremely reliable obstacle-avoiding robots that never hit anything.

For **QTb**, successful runs learned to rely more on the bump sensors once the learning algorithm converged. The rewards tended to remain high because in spite of relying upon the bump sensors, the robot still spent very little time within 20 cm of an obstacle. An inspection of the ideal inputs for the SOM for **QTb** provides a clear rationale for this behavior. There were several output nodes whose ideal inputs included high values for the bump sensors, and it was for those states that Q-learning predicted a high reward for a turn. We believe that the different patterns of SOM learning between the **QT** and **QSOM** runs is explained by the high, non-decreasing learning rate employed by **QT**.

For **Qa** and **Qb**, reliance on the sonar vs. the bump sensors varied significantly between runs. The higher-scoring runs tended to rely more on the sonar; the lower-scoring runs relied more on the bump sensors.

In all cases, poorly-performing controllers exhibited one of two behaviors: They either learned to avoid obstacles by perpetually turning, or they alternated between each of the three actions in turn. In both cases, the robots avoided hitting things largely by going nowhere. Since the reward for non-forward driving when not near an obstacle is 0.5, it is to this reward value per iteration that the poor performers tended to converge. So after a fashion, even the unsuccessful runs learned to avoid obstacles; they failed to learn to actually go anywhere while doing so, however.

CONCLUSION

We have described an empirical comparison between traditional Q-learning and two different formulations of Q-learning with the Self-Organizing Map. Our implementation employs the Lego Mindstorms NXT robot, a common platform used in undergraduate education. We have shown that the successful results from the research literature for combining these approaches can be replicated using the NXT platform as long as the motor speeds are not included as part of the input vector for the SOM.

Our results leave open a number of questions. It remains puzzling as to why the inclusion of the motor speeds in the input vector pollutes the SOM so badly in the case of **QSOMa**, as well as why **QTa** performs so badly in spite of avoiding this problem. This also raises the question of the scalability of the approach with increasing numbers of sensor inputs. We plan to experiment with multiple sonar inputs to investigate the possibility of scaling in further depth.

Another potential direction for future work would be to replicate Touzet's attempt to improve robot learning by disregarding reinforcement learning entirely [4]. In this new approach, instead of a reward function, a goal state is specified, and the robot uses the ideal inputs of the SOM to guide its actions. A second SOM encodes a relationship between actions and transitions. In light of our problems with motor speed encoding, it will be an interesting challenge to try to replicate this result on the NXT as well.

We will mention in closing that we have used QSOM as an assignment in an upper-level undergraduate Artificial Intelligence course. Our students (who, in previous assignments, had already implemented both the SOM and Q-learning separately from

each other) were very successful in implementing this combination for several tasks, including obstacle avoidance, pursuit, and light-finding.

REFERENCES

[1] Kohonen, T. *Self-Organizing Maps*, 3rd Edition. Springer, 2001.

[2] A.J. Smith. "Applications of the Self-Organizing Map to Reinforcement Learning". *Neural Networks* 15:1107-1124, 2002.

[3] C. Touzet. "Neural Reinforcement Learning for Behavior Synthesis". *Robotics and Autonomous Systems* 22(3-4):251-281, December 1997.

[4] C. Touzet. "Modeling and Simulation of Elementary Robot Behaviors using Associative Memories". *International Journal of Advanced Robotic Systems* 3(2):165-170, June 2006.

[5] C.J. Watkins and P. Dayan. "Q-Learning". *Machine Learning* 8(3-4):279-292, 1992.

[6] S. Mahadevan and J. Connell. "Automatic Programming of Behaviour-Based Robots Using Reinforcement Learning". In *Proceedings of the National Conference on Artificial Intelligence*, 768-773, 1991.

[7] F. Mondada, E. Franzi & P. Ienne, "Mobile Robot Miniaturisation: A Tool for Investigation in Control Algorithms," *Third International Symposium on Experimental Robotics*, Kyoto, Japan, October 1993.

[8] Lego Education. "Lego Mindstorms Education Base Set". http://www.legoeducation.us/store/detail.aspx?ID=1263&bhcp=1, retrieved December 11, 2009.

[9] LeJOS, Java for Lego Mindstorms. http://lejos.sourceforge.net/, retrieved December 11, 2009.

[10] Ferrer, G. *Introduction to Robotics using Lego Mindstorms NXT.* Lulu.com, 2009.

BAYESIAN DATA FUSION FOR SMART ENVIRONMENTS with HETEROGENOUS SENSORS*

Soukaina Messaoudi, Kamilia Messaoudi and Serhan Dagtas
Department of Information Science
University of Arkansas at Little Rock
2801 S. University Avenue
Little Rock AR 72204
sxmessaoudi@ualr.edu, kxmessaoudi@ualr.edu, sxdagtas@ualr.edu
(501)683-7267

ABSTRACT

Smart environments refer to buildings or locations equipped with a multitude of sensors and processing mechanisms for improved security, efficiency or functionality. Often, these sensors serve distinct purposes and their data may be processed separately by entirely separate systems. We argue that integrated processing of data available from multiple types of sensors can benefit a variety of decision making processes. For example, smart building sensors such as occupancy or temperature sensors used for lighting or heating efficiency can benefit the security system, or vice versa. Recent industry standards in sensor networks such as ZigBee make it possible to collect and aggregate data from multiple, heterogeneous sensors efficiently. However, integrated information processing with a diverse set of sensor data is still a challenge. We provide an information processing scheme that offers data fusion for multiple sensors such as temperature sensors or motion detectors and visual sensors such as security cameras. The broader goal of multi-sensor data fusion in this context is to enhance security systems, improve energy efficiency by supporting the decision making process based on relevant and

---

The research in this paper was supported in part by the US Air Force Research Laboratory Sensors Laboratory and Human Effectiveness Directorates. Some of the research activities were carried out with the technical support of Tec^Edge Discovery Laboratory at the Wright Brothers Institute in Dayton, OH.

accurate information gathered from different sensors. In particular, we investigate a major data fusion technique, Bayesian network, and present a simulation tool for a "smart environment". In addition, we discuss the potential impact of data fusion on the processes of decision or detection, estimation, association, and uncertainty management.

Key Words: Data and information fusion, Bayesian, Dempster-Shafer, Fuzzy logic, Neural Networks, Visual sensors, Non-visual sensors, Sensor networks, Motion segmentation, OpenCV.[1]

INTRODUCTION

One of the outcomes of data fusion is the improved information quality that assists various decision making processes in a "smart environment". Our focus here is the integration of sensors information into the real-time decision making process in a surveillance context. We use data fusion in a fashion where different types of information are collected from a heterogeneous set of visual and non-visual sensors. The process of integrating data from different sources requires designing an appropriate data fusion model that would take the sensor data, integrate them following a certain model, and transform it to a set of useful and relevant decisions. The anticipation is for the resulting decisions to be more accurate and efficient than those resulting from a single source. In a broader sense, we expect data fusion to lead to a virtual collaboration between the different collected information.

Towards this goal, we first investigate the usefulness of data fusion in a smart environment equipped with visual and non-visual sensors and design a convenient data fusion model. Then, we provide an overview of data fusion methods, present our data fusion algorithm and discuss our data fusion engine. This is followed by a description of our smart environment simulation tool which is used to test some of the hypotheses, visualize the environment with the sensors and their spatial relationships and to allow us to build some of the case scenarios which is discussed last. In the last section, we summarize our findings and conclusions with a set of ideas for ongoing work.

TECHNIQUES FOR DATA FUSION

Data fusion is "the theory, techniques and tools which are used for combining sensor data, or data derived from sensory data, into a common representational format." Fusing data from different sources can improve the quality and the utility of information and help improve efficiency, security and functionality. The critical problem in multi-sensor data fusion is to determine the best procedure for combining information from different sensors in the system.

Most of the reported work in data fusion uses a statistical approach in order to describe different relationships between sensors taking into account the underlying uncertainties [4]. Edward Waltz and James Llinas summarize the methods to implement data fusion as follows: decision or detection, estimation, association, and uncertainty management theories. In decision or detection theory "measurements are compared with alternative hypotheses to decide which ones best describe the measurement." Basically,

the decision theory assumes "the probability descriptions of the measurement values and prior knowledge to compute a probability value for each hypothesis." [2].

Fuzzy logic, neural networks, Bayesian, and Dempster-Shafer theories are the most commonly used methods in multi-sensor data fusion. However, our approach will focus on Bayesian model for integrated information processing using data from multiple, heterogeneous sensors. The main reasons for this election were the appropriateness of the input and output types in Bayesian model and its wide-spread use for similar problems in the literature. We plan to expand our work into the alternative fusion techniques as part of our ongoing research.

The basic principle of Bayesian theory is that all the unknowns are treated as random variables and that the knowledge of these quantities can be represented by a probability distribution. In addition, Bayesian methodology claims that the probability of a certain event represents the degree of belief that such an event will happen. The degree of belief is associated with a probability measure that can be updated by additional observed data. All the new observations are added to update the prior probability and therefore obtain a posterior probability distribution [3].

BAYESIAN DATA FUSION

The Bayesian model integrates data, independently, from r correlated sensors' inputs in the following pattern:

$$p(D / X_1^1 X_1^2 \dots X_1^r) = \frac{\prod_{j=1}^{r} p(D / X_1^j) * p(D / X_0^1 X_0^2 \dots X_0^r)}{\prod_{j=1}^{r} p(D / X_0^j)} * K$$

where K is the Bayesian normalization and is equivalent to $\frac{\prod_{j=1}^{r} p(x_1^j / X_0^j)}{p(x_1^1 x_1^2 \dots x_1^r / X_0^1 X_0^2 \dots X_0^r)}$ and $p(D / X_1^1 X_1^2 \dots X_1^r)$ is the probability of event D given $X_1^1, X_1^2, \dots, X_1^r$.

x_1^j : Current measurement/observation from correlated sensors j where j = 1, 2, …,r.

X_0^j: Prior information or old data set from correlated sensors j where j = 1, 2, …,r.

X_1^j: Posterior information or new data set from correlated sensors j where j = 1, 2, …,r.

D : Event in question (one of the decisions labeled on figure1).

The fusion engine in this project is the model we use to integrate information from both visual and non-visual sensors. The engine we design receives inputs from both visual and non-visual sensors and provides a set of relevant decisions (outputs).

As the diagram in Figure 1 shows, $s_1, s_2, s_3 \dots s_n$ are inputs from different non-visual sensors. These inputs first go through a correlation model (raw data processing on figure1) that determinates the correlations among the sensors' inputs and transmits

independent m outputs that are fed to the fusion engine as inputs. These outputs (fusion engine inputs) are labeled as $x_1, x_2, x_3, \ldots x_m$.

The fusion engine inputs $x_1, x_2, x_3, \ldots x_m$ can be matched to notations such as $X_1^1, X_1^2, X_1^3 \ldots, X_1^r$, which represent the posterior information, described in the algorithm section, from correlated sensors. However, this matching does not restrict matching x_1 to X_1^1, x_2 to $X_1^2, \ldots$ etc as the data fusion model we use consider integrating posterior information from both non-visual and visual sensors. As it is explained below, data from visual sensors is pre-processed before it can be fed to the fusion engine. This pre-processing results in a convenient format of information to be passed to the fusion engine.

For visual sensors, we use optical and infrared cameras to record raw videos. The acquired videos are then processed to extract meta-data information to be used in the fusion algorithm described above. The processing of images from such visual sensors requires a preliminary processing where some intermediary image features such as moving objects and their boundaries are extracted for further processing [5]. The final extracted visual information forms metadata that can be fed to the designed fusion engine that integrates it with other sensor data from other heterogeneous sensors.

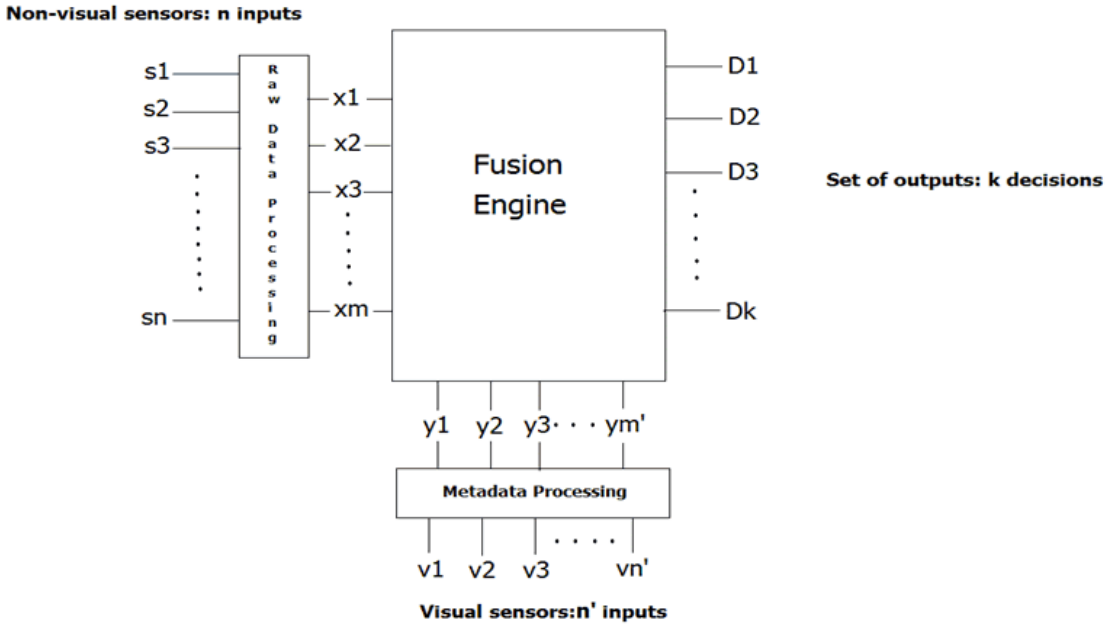


Figure1: Fusion Engine Design

The extraction of visual information can be a real challenge because of "the lack of proper low-level algorithms for robust feature extraction" [7]. Here, we use a motion detection algorithm to extract relevant visual information about the moving objects in the recorded video. The algorithm chosen for this purpose is the implementation in OpenCV, which is an open-source computer vision library, originally developed by Intel. We have performed a few modifications at the input level that resulted in movement detection. The metadata in this context includes the kind of information such as the number of moving

objects, the nature of movement, the type of the moving objects (human or animal), the actions performed by the moving objects, the area they occupy, and the time they stay in the room of question.

In the fusion engine design on Figure1, $v_1, v_2, v_3, \ldots v_n$, represent the information collected (metadata) from the every visual sensor (n' visual sensors). These inputs are processed (metadata processing in Figure 1) to create appropriate input format. The resulting outputs of the metadata processing are also in the form of correlated information. In other words, some visual sensors can be correlated in the sense that only one output can be retrieved from them. This correlation of visual sensors results in independent inputs labeled as $y_1, y_2, y_3, \ldots y_m$, in Figure 1.

After tracking moving objects on a given video, more work is done on detecting the different features of these moving objects. Features such as the number of moving objects, the nature of the moving objects (human, animal…), and the nature of movements (fast, slow…) the objects perform are examples of information we want to feed to the fusion engine. After extracting such important information (metadata), we perform another processing on the metadata to come up with an input format compatible with the data fusion model we are using (Bayesian model).

In data fusion context, the outputs of such a model are in the form of decisions that should be performed to better serve the environment where the different types of sensors are used. As Figure 1 shows, the set of decisions $D_1, D_2, \ldots, D_k$ are the independent fusion engine outputs (or decisions). These decisions can help in saving energy, restricting security, launching rescue operations and many more. Depending on what type of sensors we use, a set of relevant and efficient decisions can be formed.

SIMULATION TOOL & EXPERIMENTS

In our study of multi-sensor data fusion, we implement a simulation tool that helps us construct a virtual smart environment. The smart environment has basically different types of sensors such as: motion detector, smoke detector, daylight sensor, and other types of sensors. In addition to sensors, there are objects that can be moving around to generate case scenarios where motion is a factor to be considered. Emergency cases such as fire or flood can be studied using the implemented simulation tool. This tool is implemented using JAVA and it facilitates the study of multiple scenarios because the user can chose any type of sensors implemented in the tool as well as manage the environment's state such as increasing the temperature (fire case) or adding moving objects or water (flood scenario). Visual sensors are placed on the simulation grid at specific grid locations. A specific set of attributes must be defined for each sensor. These may include range, angle, sensitivity, and direction. Every sensor has a detection area and detection occurs when the coverage area and attributes of a given object overlap with the detection range and sensitivities of a given sensor. The simulation tool is our main data generator where sensors' flags and data are fed to the fusion engine where decision making process takes place.

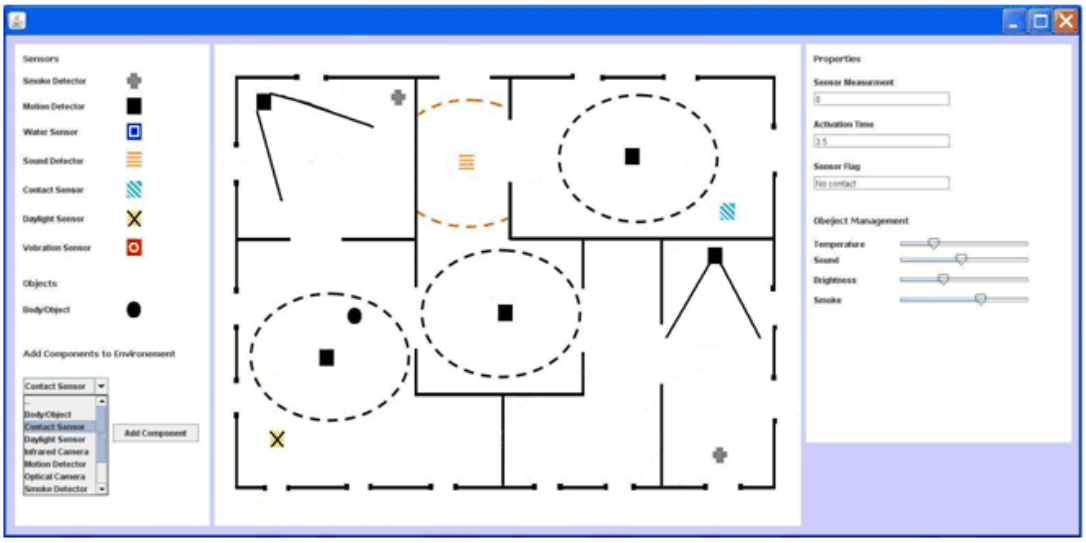


Figure 2: Simulation tool interface

In order to develop a reasonable method for finding a likelihood function at a given moment, we have carefully studied the behavior of the moving objects. We have conducted ten experiments where we tracked one object in every video and recorded the corresponding data. Because of space limitations, we present only the conclusions we have deived from the analysis of data. Through analyzing the graphs from the experiments, we take into consideration the factor of persistence, which merely means for how long the object(s) is moving. In order to do this, we choose a time instance from the plot and study the behavior of the moving object in previous time instants.

The probability value that will be used by the fusion engine at time is computed as follows:

$$y_i = \sum_{i=0}^{m-1} X(t_i).\left[\frac{1-i}{10}\right]$$ where $X(t_i)$ is t_i's equivalent area percentage.

In order to find the reasonable number of previous time instants that should be included in the computation of a given likelihood function at a given instant, we further analyze the data collected from the ten experiments. We apply the formula above at t_9 for every experiment and find the equivalent likelihood function (y_9) taking into consideration m=10, 8, 6, and 4 previous time instants. The table below summarizes the analysis:

	Exp1	Exp2	Exp3	Exp4	Exp5	Exp6	Exp7	Exp8	Exp9	Exp10
m=10	117.70%	108.40%	75.83%	210.48%	62.10%	27.29%	84.31%	195.53%	72.45%	80.17%
m=8	77.50%	77.84%	55.21%	151.95%	40.81%	20.20%	58.00%	127.93%	48.65%	51.80%
m=6	56.93%	48.38%	35.18%	95.94%	22.13%	13.10%	34.40%	76.18%	28.03%	28.60%
m=4	28.60%	24.66%	18.76%	52.01%	9.48%	6.52%	16.17%	36.50%	13.81%	12.82%

Table 1: Computed likelihood function at t_9 using the weighted method.

From the table above, we conclude that looking back at eight or six time instants usually result in a reasonable value that gives us an idea about how intense the motion is in a given room and can safely be fed to the fusion engine. Also, the computation of a likelihood function for m=8 or 6 is easy and quicker than m=10 or more; it also doesn't take into consideration the percentage value at where usually no motion is recorded.

CONCLUSIONS

We have demonstrated ways to use Bayesian data fusion technique in a smart environment with a heterogeneous, inter-dependent set of sensors. This was done by generating statistically independent inputs for the Bayesian fusion model and demonstrate the effect through a simulation tool. The Dempster-Shafer theory is considered to be a generalization of the Bayesian theory of subjective probability. Dempster-Shafer allows us to "base degrees of belief for one question on probabilities for a related question" [6]. One of the most important advantages of the Dempster-Shafer theory is that it does not associate probabilities to questions of interest as Bayesian methods do. Instead, the belief for one question is based on probabilities for a related question; therefore, the Dempster-Shafer theory can effectively model uncertainty. As a next step, we plan to build a Dempster-Shafer model and draw comparisons with the Bayesian model. Additionally, further experimentation is underway using a testbed created by ZigBee-based sensors that implement the smart environment and by optical and infra-red cameras.

REFERENCES:

[1] Challa, S., Koks, D., An Introduction to Bayesian and Dempster-Shafer Data Fusion, http://robotics.caltech.edu/~jerma/research_papers/BayesChapmanKolmogorov.pdf, retrieved October 28, 2009.

[2] Waltz, Edward, James, L. ,Multisensor Data Fusion. Norwood: Artech House, Inc, 1990.

[3] Fisher C., Eitel L., Richard W., and Shobha C., Introduction to Information Quality. Cambridge : MITIQ, 2006.

[4] H.B, Mitchell, Multi-Sensor Data Fusion. Berlin: Springer, 2007.

[5] Snidaro, L., Claudio, P., Christian, M.,Gian, F., Visual Sensor Technology for Advanced Surveillance Systems: Historical View, Technological Aspects and Research Activities in Italy, http://www.mdpi.com/1424-8220/9/4/2252/pdf, retrieved November 12, 2009.

[6] Shafer, G., Dempster-Shafer Theory, http://ww

ANOMALY DETECTION OF MASQUERDERS BASED UPON TYPING BIOMETRICS AND PROBABILISTIC NEURAL NETWORK*

Yingbing Yu
Computer Science & Information Technology Department
Austin Peay State University
Clarksville, TN 37044
931 221-7826
yuy@apsu.edu

ABSTRACT

This paper investigates a new method to more effectively detect anomaly intrusions from masqueraders. Typing biometrics of keystroke patterns from users are collected and used as the normal behavior profiles. The supervised learning probabilistic neural network (PNN) is used for the classification of data as from a normal user or from a masquerader. Experimental show the good results with a high rate of successful detection of masqueraders and a low false alarm rate.

I. INTRODUCTION

IDSs attempt to perform the process of monitoring computer networks and systems for violations of security policy (a set of laws, rules, and practices that define the system boundaries) [1-2]. IDSs can be categorized into misuse two classes based on different detection approaches. Misuse (knowledge or signature-based) IDSs look for specific patterns that define a known attack. The information about known attacks and vulnerabilities of a system is encoded into "signatures". Any actions that trigger the matches will be reported as "attempts" of intrusions. Anomaly (behavior-based) IDSs assume the deviation of normal activities under attacks and perform abnormal detection compared with predefined system or user behavior profiles.

Anomaly IDSs can be used to detect inside attacks from masqueraders, defined as internal or external intruders who exploit legitimate users identification and password obtained illegally to perform malicious attacks. Inside abuse of computer system was reported as the second most cited forms of attacks which contributed to a large portion of financial loss [3]. To prevent a system from attacks due to identity theft, the effective approach is to deploy effective anomaly IDS to monitor user behavior and report any suspicious activities. Alarms are reported when an acclaimed user (masquerader) behaves out of characters and a large deviation with the genuine user's behavior profile is detected. To distinguish a masquerader from genuine users is a challenging task due to the problem of concept drift, where the observed user behavior may change with different tasks, time, general knowledge level and such other uncertain elements [4].

In this paper, we introduce a model to detect masqueraders using the typing biometrics of keystroke patterns. The rest of this paper is organized as follows. Section II is the literature review that discusses masquerader detection using command sequences and typing biometrics. Section III presents the user typing biometrics template and the classification of patterns using probabilistic neural network (PNN). Section IV presents the experimental results as the detection rate of masquerader and false alarms of incorrect classification. The paper concludes with section V, which discusses the future research work.

II. LITERATURE REVIEW

Access control and authentication are not sufficient to prevent potential intrusions from masquerader which already got the authorization to access system resources by obtaining an authorized user identity illegally. User behavior profiling can be used for the purpose of classification, future behavior prediction and masquerader detection. Traditionally user behavior in a system is characterized by parameters such as login frequency, location frequency, last login, session elapsed time, password fails, location fails, amount of network traffic, resources used by user in a session and so on [5].

De Ru etc. developed a software methodology that improves security by using typing biometrics to reinforce password authentication mechanisms [12]. Typing biometrics is the analysis of a user's keystroke patterns. Each user has a unique way of using the keyboard to enter a password. For example, each user types the characters that constitute the password at different speeds. The methodology employs fuzzy logic to measure the user's typing biometrics.

Machine learning and statistical methods have been widely used for the behavior profiling from the analysis of command sequences. Davison and Hirsh developed a model called IPAM (incremental probabilistic action modeling) to predict sequences of user actions [6]. Single-step command transition probability is estimated from training data. Balajinath introduced a Genetic Based Intrusion Detector (GBID) to model individual user behavior with a 3-tuple vector which is learnt later via a genetic algorithm [11]. Ryan used a back propagation neural network NNID (Neural Network Intrusion Detector) to identify users simply by what commands and how often they use, called the 'print' of a user [7].

Lane and Brodley [8] chose a machine learning algorithm IBL (instance based learning) to measure the similarity between the most recent 10 commands of a user and the profile extracted from the past. The similarity measure is the count of matches of a new sequence with the sequences from a user's command history, with a greater weight assigned to adjacent matches. Schonlau selected several statistics-based methods to detect masqueraders, including uniqueness, Bayes one-step Markov, Compression, Multi-step Markov chain etc [9]. Maxion and Townsend applied Naïve Bayer classification algorithm to user profiling with command-line data [10], which shows improvement over the best approach of Schonlau.

III. MASQUERDER DETECTIN FROM TYPING BIOMETRICS

Each user types the characters that constitute a command at different speeds. Typing biometrics can be used for the analysis of a user's keystroke patterns. Our method is to build a biometrics template served as the behavior profiling to be used for the anomaly detection. In this approach, when an authorized user is accessing a system and typing commands, the interval time in CPU cycles between two successive characters in a unique command is recorded. For example, if the user types a command "dir", the time intervals between the characters "d" and "i", "i" and "r", "r" and the "ENTER" key will be stored as the biometric characteristic.

If the command contains parameters, the time between the space and a regular character is also recorded. For example, if a user types the command "dir –p", we will also calculate the time interval between the characters of "r" and the "BLANK" key. A user may type capital letters instead of lower case ones, and this will involve a larger time interval since the user needs to press the "CapsLock" key at the same time. We have not considered this scenario and also experiments show that most users prefer to always type commands in lower case. In addition, we only consider those frequently typed commands by a user. One reason is that later a neural network is used for the classification and a relatively large training data set is required.

The user typing biometrics in the form of time interval of successive characters is used as the typing template for the user. On subsequence access to the system, each of the user command typing will go through a neural network system for the classification as normal from the genuine user or abnormal from a masquerader. We have chosen the supervised probabilistic neural network (PNN) for the classification purpose. In the following section, we give a brief overview of PNN and how it can be applied for the user type biometrics to do the classification as a genuine user or masquerader.

Probabilistic Neural Network for Classification

Probabilistic neural network, which was first proposed by D. F. Specht [13], is a kind of supervised neural network that has been widely used in the area of pattern recognition, nonlinear mapping, and classification. It consists of three feed-forward layers: input layer, pattern layer, and summation layer. An input layer contains as many elements as there are separable parameters needed to describe the objects to be classified. It computes distances from the input vector to the training input vectors and produces a vector whose elements indicate how close the input is to a training input. The pattern

layer sums these contributions for each class of inputs to produce a vector of probabilities as its net output. The summation layer (output layer) has as many processing elements as there are classes to be recognized. A compete transfer function on the output of the second layer picks the maximum of these probabilities, and produces a "1" for that class and a "0" for the other classes.

The pattern layer represents a neural implementation of a version of Bayes classifier, where the class nonparametric probability density functions are approximated using a Parzen estimator [14]. This approach provides an optimum pattern classifier to minimize the expected risk of wrongly classifying an object. It gets closer to the true underlying class density functions as the number of training samples increases if the training set is an adequate representation of the class distinctions. PNN has the learning and generalization ability of back-propagation multi-layer neural networks. It can capture the relationships between the training examples and given classification. PNN training scheme is simpler and faster than back-propagation networks and has the following advantages: rapid training speed; guaranteed convergence to a Bayes classifier if enough training examples are provided; enables incremental training which is fast; robustness to noisy examples [15].

In our model of the classification after the user behavior profile as the typing biometrics template is built, we pick up a block size of 10 commands for the testing. The reason is that a user session usually involved the execution of multiple commands which is reasonable to make a judgment after a certain block size of commands is executed in the system. It also prevents the possible high false alarm rate which is a major issue for anomaly intrusion detection systems. In the testing, to determine if the whole block of commands is from the genuine user or a masquerader, we have chosen a threshold value of 4 mismatched commands to classify a block as normal or abnormal. If 4 or more commands in a block are classified as abnormal, the whole block is labeled as abnormal and accordingly the user is regarded as the possible masqueraders for further investigation to issue security alerts.

IV. EXPERIMENTAL RESULTS

We have tested the model based on typing biometrics and probabilistic neural network to detect masqueraders by using the data generated in a computer networking system. Ten authorized users have participated in the experiment to collect the data during the training period to build the behavior profile separately. All the data was generated when they logged on the computer system using the assigned user id and password. In addition, to generate the masquerader data, another 5 different users from the outside participated to generate the test data when they work on the same computer system.

For each of the 10 normal users, the generated training data is 1200 commands in which the first 1000 commands are used for the training and the other 200 for the testing purpose. For each of the other 5 outside users, each also generated a data set of 200 commands. All the data from the 5 outside users is used as the masquerader data against each of the normal users. As discussed in the last section, all the testing data of user commands is divided into blocks of 10 in each. For each of 10 normal users, the 20 blocks

of normal testing is to check whether the model can correctly label it as "normal" and otherwise it triggers a false alarm. The data of 100 blocks in total from the 5 outside users is to test if the model can correctly identify anomaly cases.

Table 1 shows the results of successful masquerader detection rate and false alarm rate when the normal cases from the genuine user are classified incorrectly as "abnormal". For most users, the model can achieve a high masquerader detection rate. The average detection rate for the 10 users is about 89.4% (894/1000) and the missing percentage is 10.6% (106/1000). If a normal one is classified incorrectly as abnormal, a false alarm is generated. In the experiment, only 16 of 200 normal cases are incorrectly identified as "abnormal" and the false alarm rate is 8.0%.

Table 1. Anomaly detection and false alarm rate

	Detection Rate	False Alarm Rate
User 1	91.0% (91/100)	5.0% (1/20)
User 2	85.0% (85/100)	0.0% (0/20)
User 3	95.0% (95/100)	10.0% (2/20)
User 4	91.0% (91/100)	15.0% (3/20)
User 5	86.0% (86/100)	5.0% (1/20)
User 6	87.0% (87/100)	0.0% (0/20)
User 7	88.0% (88/100)	10.0% (2/20)
User 8	93.0% (93/100)	10.0% (2/20)
User 9	91.0% (91/100)	15.0% (3/20)
User 10	87.0% (87/100)	10.0% (2/20)
Average	89.4% (894/1000)	8.0% (16/200)

We further conducted experiments with different block sizes to compare the performance of detection and false positive alarm rate. The results in Table 2 show the comparison results of different block sizes. For example, if the block size is 5 commands, the testing cases are doubled with 2000 blocks from masqueraders and 400 normal testing blocks in total. For the block size of 5, the detection rate is 85.6% and false alarm rate is 10.3% (400 anomaly test cases). If a block size of 15 commands is chosen, the detection rate is 90.5% and false alarm rate is 10.0%. A block size of 20 and 25 commands is also chosen and the results is listed in the table.

Table 2. Anomaly detection and false alarm rate of different block sizes

Block Size (# of commands)	Detection Rate	False Alarm Rate
5	85.6% (1712/2000)	10.3% (41/400)
10	89.4% (894/1000)	8.0% (16/200)
15	90.5% (588/650)	10.0% (13/130)
20	89.2% (446/500)	12.0% (12/100)
25	86.0%(344/400)	16.3% (13/80)

For a real intrusion detection system, it is the ultimate goal to detect masqueraders within a short time interval and alert the system earlier to prevent further loss. Based on the experimental results, the interval of 10 commands execution achieves both a high detection rate and a low false alarm rate. If a larger test block size is selected, the model can still achieve a high detection rate but in the same time it introduces a higher false alarm rate. If the block size is large above a threshold value (25 commands in the experiment), the performance of the model decreases steadily with a low detection rate and high false alarm. One reason is that the probability of overlapping behavior patterns increases rapidly when the target size is above a certain threshold. In practice, an appropriate size can be selected based upon specific security policies for an organization. In general, it is fairly reasonable and effective for an anomaly IDS to detect potential masqueraders after just about 10 or 15 commands execution.

V. CONCLUSIONS

In this paper, we introduce the neural network to detect masquerades based on user behavior profiling in anomaly intrusion detection systems. Typing biometrics of keystroke patterns from users can be collected in the training phase as the biometric templates. To do the classification of a typing pattern, the supervised learning neural network PNN is used considering its effectiveness in the areas of pattern recognition, nonlinear mapping, and classification. PNN has the learning and generalization ability of back-propagation multi-layer neural networks (BPNN) and is simpler and faster. It can identify the commonalities in the training examples and then perform classification of unseen examples from the predefined classes. Experiments conduced in a real computer environment show promising results.

We want to extend the current research of masquerader detection. As we have notices that in the last decade GUI and Internet-based applications have been deployed in both UNIX and Windows systems. A large of part of user activities associated with these applications may not involve individual commands directly entered into the system, but instead consist of mouse clicks on icons. The behavior modes from this kind of activity will differ significantly from those discussed in this paper. Future work would address these questions.

REFERENCES

[1] Anderson, James P., "Computer Security Threat Monitoring and Surveillance," Technical Report, James P. Anderson Co., Fort Washington, Pa., 1980.

[2] Bace, R., "Intrusion Detection," Macmillan Technical Publishing, 2nd Edition, Indiana, 2000.

[3] Computer Security Institute and Federal Bureau of Investigation, "2006 CSI/FBI Computer Crime and Security Survey," Computer Security Institute publication.

[4] Lane, Terran, "*Machine Learning Techniques for the Computer Security Domain of Anomaly Detection",* PhD thesis, Purdue University, W. Lafayette, IN, August 2000.

[5] Denning, Dorothy E., “An Intrusion Detection Model”, IEEE Transactions on Software Engineering, 13(2): 222–232, February 1987.

[6] Davison, B. and Hirsh, H., “Predicting sequences of user actions,” Predicting the Future: AI Approaches to Time-Series Problems, 1998 AAAI Workshop, July 1998, Madison, Wisconsin, pp. 5–12.

[7] Ryan, J., Lin, M., Miikkulainen, R., “Intrusion Detection with Neural Networks”, Advances in Neural Information Processing Systems 10, M. Jordan et al., Eds., Cambridge, MA: MIT Press, 1998, pp. 943-949.

[8] Lane, T. and Brodley, C. E., “*Temporal Sequence Learning and Data Reduction for Anomaly Detection”*, *ACM Transactions on Information and System Security,* 2(3). Pp. 295—331, 1999.

[9] Schonlau, M., DuMouchel, W., Ju, W., Karr, A., Theus, M., Vardi, Y., “Computer Intrusion: Detecting Masquerades”, Statistical Science, 16(1): 58-74, 2001.

[10] Maxion, Roy and Townsend, Tahlia. “Masquerade Detection Using Truncated Command Lines”, International Conf. on Dependable Systems&Networks: Washington, DC, 23-26 June 2002: 219-228.

[11] Balajinath, B., Raghavan, S. V., “Intrusion detection through learning behavior model”, Computer Communication 24(2001) 1202-1212.

[12] De Ru, Willem G., Eloff, Jan H.P., “Enhanced Password Authentication through Fuzzy Logic,” IEEE Expert, Volume 12, Issue 6, Nov.-Dec. 1997 Page(s):38 – 45.

[13] Specht, D.F., “Probabilistic Neural Networks”, Neural Networks, vol. 3, pp.109-118, 1990.

[14] Parzen, E., "On estimation of a probability density function and mode", Ann. Math. Statist., vol. 33, pp. 1065-1076, 1962.

[15] Wasserman, P.D., “Advanced Methods in Neural Networks”, Van Nostrand Reinhold, New York, pp.35-55, 1993.

DJANGO, A WEB FRAMEWORK USING PYTHON*

TUTORIAL PRESENTATION

Carl Burch
Hendrix College, 1600 Washington Ave, Conway AR 72032
(501) 450-1377
cburch@cburch.com

OVERVIEW

Web developers often use a Web application framework for building applications that interface with Web and database servers. Some of the more popular frameworks and their associated languages include Rails (Ruby), Spring MVC (Java), Struts (Java), and Pylons (Python). Yet despite their importance in industry, such frameworks are often omitted from the undergraduate computer science curriculum.

This tutorial will explore a particularly popular Web application framework for Python called Django [1,2]. Like most Web application frameworks, Django provides much more support for Web applications than older CGI libraries. Through a templating system, developers can separate application logic from the user interface implemented in HTML. Django also provides a Python-based API for defining and accessing a database, automating the creation of SQL statements and conversion of data for simpler queries (though still allowing SQL to be used for more complex queries). Similarly, it includes an API for generating HTML forms and retrieving data from them. And it includes logic for common Web application features, such as session management, cookie access, and user logins.

The tutorial will describe the more prominent features of Django, and it will include reflection on the presenter's experience with using Django when teaching it to upper-division computer science students.

PRESENTER

Carl Burch is an Associate Professor of Computer Science at Hendrix College. He regularly teaches an upper-division course on databases, which includes a heavy emphasis on Web development – and on Django in particular during the most recent iteration.

REFERENCES

[1] Django home page, http://www.djangoproject.com/, retrieved November 24, 2009.

[2] Holovaty, A., Kaplan-Moss, J., *The Definitive Guide to Django: Web Development Done Right*, Berkeley, CA: Apress, 2009.

Papers of the Sixteenth Annual CCSC Central Plains Conference

April 9-10, 2010
Park University
Parkville, Missouri

WELCOME — 2010 CCSC: CENTRAL PLAINS CONFERENCE

Park University is pleased to host the Sixteenth Annual Consortium for Computing Sciences in Colleges Central Plains Conference in 2010. The conference this year presents an interesting and varied program, including software engineering, database, networking, security, computer architecture, discrete mathematics, active learning, capstone experiences, and features a keynote address by the founder of Notable Software, Inc. The tutorial program includes a stimulating variety of topics including distributed system, version control, and automatic grading system. The program also offers sessions for "nifty" assignments and a pre-conference workshop entitled "JGrasp: An integrated development environment with intuitive visualizations for teaching hard concepts in Java." The conference also continues to host student activities. This year we have a student web site contest, a student programming contest and a student poster contest. Cash prizes are available for all student contests.

Published materials for this conference were selected using a double-blind refereeing process, with reviewers solicited from the ACM SIGCSE paper submission web site. Our thanks go to Phil Heeler and Henry Walker for configuring this system and supporting its use. Over 100 reviews were completed by the reviewers during the paper selection process. In 2010, 65% of the regular papers submitted were accepted for publication in this volume of *The Journal of Computing Sciences in Colleges* and for presentation at the conference.

On behalf of the Regional Board and the Conference Steering Committee, I wish to thank all those who have contributed to the success of this conference: the authors for their submissions, the reviewers, the session chairs and all those who took leadership roles on the steering committee. I would also like to thank Park University, especially the administrative staff, and my colleagues in the Information and Computer Science Department who helped make the conference a success. Finally, thanks to the many individuals, vendors, and organizations whose support, either financially or through volunteering time and other resources, helped make the conference possible.

If you have questions or comments during or after the conference, please contact any Central Plains Regional Board or Conference Steering Committee member. We hope you enjoy the 2010 CCSC Central Plains Conference and the time you spend at Park University in the small and beautiful town of Parkville.

Wen-Jung Hsin
Park University
Conference Chair

2010 CENTRAL PLAINS CCSC REGIONAL BOARD

MEMBERS

Scott Sigman, Regional Representative and Board Chair (2011) . . . . . Drury University, Springfield, MO
Gary Schmidt, Registrar and Membership Chair (2011) . . . . . Washburn University, Topeka, KS
Judy Mullins, Treasurer (2012) University of Missouri Kansas City, Kansas City, MO
Dean Sanders, Editor (2012) . . Northwest Missouri State University, Maryville, MO
Cecil Schmidt, Secretary (2010) . . . . . Washburn University, Topeka, KS
Chuck Pheatt, Webmaster (2010) . . . . . Emporia State University, Emporia, KS
Tim DeClue, Past Conference Chair (2010) . . . . . Southwest Baptist University, Bolivar, MO
Wen Hsin, Current Conference Chair (2010) . . . . . Park University, Parkville, MO
Mahmoud Yousef, Next Conference Chair (2011) . . . . . University of Central Missouri, Warrensburg, MO

2010 CENTRAL PLAINS CCSC CONFERENCE STEERING

COMMITTEE

Wen Hsin, Conference Chair . . . . . Park University, Parkville, MO
Phil Heeler, Papers . . . . . Northwest Missouri State University, Maryville, MO
Dean Sanders, Regional Editor Northwest Missouri State University, Maryville, MO
Ron McCleary, Panels, Tutorials and Workshops Avila University, Kansas City, MO
John Cigas, Student Programming Competition . . . . Park University, Parkville, MO
Jim Cain, Student Programming Competition . . . . . Southwest Baptist University, Bolivar, MO
Scott Bell, Student Programming Competition . . . . . Northwest Missouri State University, Maryville, MO
Brian Hare, Student Programming Competition . . . . . University of Missouri-Kansas City, MO
Carol Spradling (CoChair), Student Web Page Contest . . . . . Northwest Missouri State University, Maryville, MO
Scott Sigman (CoChair), Student Web Page Contest . . . . . Drury University, Springfield, MO
Ed Mirielli, Student Web Page Contest . . . . . Westminster College, Fulton, MO
Rick Barker, Student Posters . . . . . Washburn University, Topeka, KS
Mike Beneke, Student Posters . . . . . Westminster College, Fulton, MO

Mahmoud Yousef, Nifty Course Assignments . . . . . University of Central Missouri, Warrensburg, MO
George Gibeau, Nifty Course Assignments . . . . . Ozarks Technical Community College, Springfield, MO
Rad Alrifai, Nifty Course Assignments . . . . . Northeastern State University
Gary Ury, Lightning Talks . . . . Northwest Missouri State University, Maryville, MO
Carol Browning, Vendors - Grad School Liaison . Drury University, Springfield, MO
Gary Schmidt, Registrar - Regional . . . . . Washburn University, Topeka, KS
Wen Hsin, Registrar - Local . . . . . Park University, Parkville, MO
Scott Sigman, National Representative . . . . . Drury University, Springfield, MO
Cecil Schmidt, Secretary . . . . . Washburn University, Topeka, KS
Judy Mullins, Regional Treasurer . . . . . University of Missouri - Kansas City, MO
Wen Hsin Park, Local Arrangements, Publicity . . . . Park University, Parkville, MO
Chuck Pheatt, Webmaster . . . . . Emporia State University, Emporia, KS

REVIEWERS — 2010 CCSC CENTRAL PLAINS CONFERENCE

Mohammad Alanazi . . . . . Imam University, Riyadh, Saudi Arabia
Rad Alrifai . . . . . Northeastern State University, Tahlequah, OK
Khaled Alshare . . . . . Emporia State University, Emporia, KS
Pavel Azalov . . . . . Penn State University, Hazelton, PA
Beverly Bohn . . . . . Park University, Parkville, MO
Nick Breems . . . . . Dordt College, Sioux Center, IA
Carol Browning . . . . . Drury University, Springfield, MO
Chia-Chu Chiang . . . . . University of Arkansas at Little Rock, Little Rock, AR
Monica Costa . . . . Polytechnic Institute of Castelo Branco, Castelo Branco, Portugal
Tim DeClue . . . . . Southwest Baptist University, Bolivar, MO
George Dlmitoglou . . . . . Hood College, Portland, OR
Jeffrey Edgington . . . . . University of Denver, Denver, CO
Emanuel Emanouilidis . . . . . Kean University, Union, NJ
Ernest Ferguson . . . . . Northwest Missouri State University, Maryville, MO
Charles Frank . . . . . Northern Kentucky University, Highland Heights, KY
Ernie Giangrande Jr. . . . . Wingate University, Wingate, NC
Nadeem Hamid . . . . . Berry College, Mount Berry, GA
James Harris . . . . . Pittsburg State Univ, Pittsburg, KS
Phillip Heeler . . . . . Northwest Missouri State University, Maryville, MO
Dennis Herr . . . . . Missouri Southern State University, Joplin, MO
Wen-Jung Hsin . . . . . Park University, Parkville, MO
Myungsook Klassen . . . . . California Lutheran University, Thousand Oaks, CA
Janet Kourik . . . . . Webster University, St. Louis, MO
Srinivasarao Krishnaprasad . . . . . Jacksonville State University, Jacksonville, AL

Noel LeJeune . . . . . Metropolitan State College of Denver, Denver, CO
Hong Lin . . . . . University of Houston-Downtown, Houston, TX
Hui Liu . . . . . Missouri State University, Springfield, MO
Ron McCleary . . . . . Avila University, Kansas City, MO
Robert McCloud . . . . . Sacred Heart University, Fairfield, CT
Jim McKeown . . . . . Dakota State University, Madison, SD
Bruce Mechtly . . . . . Washburn University, Topeka, KS
Jose Carlos Metrolho . . . . . Instituto Politenico de Castelo Branco, Portugal
Larry Morell . . . . . Arkansas Tech University, Russellville, AR
Judy Mullins . . . . . University of Missouri – Kansas City, Kansas City, MO
David Naugler . . . . . Southeast Missouri State University, Cape Girardeau, MO
Robert Neufeld . . . . . McPherson College, McPherson, KS
Chuck Pheatt . . . . . Emporia State University, Emporia, KS
David Pope . . . . . Ozarks Technical Community College
Hassan Pournaghshband . . . . . Southern Polytechnic State University, Marietta, GA
Dean Sanders . . . . . Northwest Missouri State University, Maryville, MO
Jamil Saquer . . . . . Missouri State University, West Plains, MO
Cecil Schmidt . . . . . Washburn University, Topeka, KS
Terry Scott . . . . . University of Northern Colorado, Greeley, CO
Onkar Sharma . . . . . Marist College, Poughkeepsie, NY
Ching-Kuang Shene . . . . . Michigan Technological University, Houghton, MI
Scott Sigman . . . . . Drury University, Springfield, MO
Carol Spradling . . . . . Northwest Missouri State University, Maryville, MO
Keith Tookey . . . . . Eureka College, Eureka, IL
Gary Ury . . . . . Northwest Missouri State Universiy, Maryville, MO
Ian van der Linde . . . Anglia Ruskin University, Cambridge & Chelmsford, England
Ken Vollmar . . . . . Missouri State University, Springfield, MO
Henry Walker . . . . . Grinnell College, Grinnell, IA
George Whitson . . . . . The University of Texas at Tyler, Tyler, TX
Paul Wiedemeier . . . . . The University of Louisiana at Monroe, Monroe, LA
Mahmoud Yousef . . . . . University of Central Missouri, Warrensburg, MO

KEYNOTE ADDRESS

Friday, April 9, 2010

FORENSICS IN THE COMPSCI CLASSROOM *

Rebecca Mercuri, Ph.D.
Notable Software, Inc.

Although forensics is a well-established discipline in biology, medicine, physics, materials and other sciences, its applications to computing are still relatively new. The television shows, NUMB3RS and CSI, have helped popularize the field by providing insight into algorithmic methods used to solve crimes. As it happens, forensic techniques are also useful in the exploration of a broad range of computational topics within a learning experience that emphasizes creativity in problem-solving. For example, simple, public-domain tools can be used to recover deleted data on USB thumbdrives, a process that demonstrates how different operating systems may deal with files, directories and storage media. This talk will overview the fundamentals of computer forensics, while providing a slew of examples (along with software references) that can be easily adapted to the study of Computer Science and Information systems.

Rebecca Mercuri is the lead forensic expert at Notable Software, Inc. <www.notablesoftware.com>, the company she founded in 1981. Her caseload has included matters involving contraband, child endangerment, murder, computer viruses and malware, wrongful work termination, class-action suits, copyright and patent infringement, and election recounts (most notably Bush vs. Gore). Dr. Mercuri has provided formal testimony and comment to the House Science Committee, the U.S. Commission on Civil Rights, the Election Assistance Commission, the National Institute of Standards and Technologies, the U.K. Cabinet, and numerous state legislatures and municipal bodies. She is a senior life member of the Association for Computing Machinery, where she has authored the Security Watch feature and numerous guest columns of Inside Risks for Communications magazine, and is a co-founder and past chair of the professional joint chapter of the Princeton ACM/IEEE Computer Society. Rebecca is also an adjunct member of the Computer Engineering faculty at The College of New Jersey, where she teaches a broad range of topics, including a senior engineering lecture/laboratory elective on Digital Forensics.

BANQUET ADDRESS

Friday, April 9, 2010

PLANETARY ADVENTURES: MY LIFE AS A NASA SOFTWARE ENGINEER [*]

Carol Browning

Drury University
Springfield , Missouri
cbrowning@drury.edu

There are a number of stories from experiences working at NASA that make good classroom anecdotes. These come from working in different roles on a variety of missions and pre-missions. The Telerobot Project was prototype work for a remote controlled robot, which led to applications for the robot arm on the space shuttle and the rovers on Mars. During the Voyager Neptune Encounter, a natural language processor was coupled with a database which was constantly being updated with information, such as the number of known moons of Neptune. This system was available in the press room during the encounter. The Sequence Automation Research Group applied state-of-the-art technologies to automating the sequence planning process for missions, including the Galileo Mission to Jupiter. On its way to study the polar regions of the sun, the Ulysses mission flew close to Jupiter for a gravity assist. Some of these experiences refer to planning the science activities on the Ulysses spacecraft during the Jupiter fly-by and to negotiating for Deep Space Network time for the mission. The SIR-C shuttle-based Earth imaging RADAR mission had software system engineering challenges for the mission operations system. This system involved four internal software systems and four external systems, including the flight software and the RADAR system equations software. For the Cassini mission to Saturn, these experiences include testing the flight software for the RADAR instrument, developing the instrument operations system, and completing the preliminary planning for the operations of the RADAR instrument.

---

jGRASP: AN INTEGRATED DEVELOPMENT ENVIRONMENT WITH INTUITIVE VISUALIZATIONS FOR TEACHING HARD CONCEPTS IN JAVA*

PRE-CONFERENCE WORKSHOP

James H. Cross II
Department of Computer Science and Software Engineering
Auburn University, Auburn, AL 36849
334-844-6315, crossjh@auburn.edu

jGRASP is a lightweight IDE that provides automatic generation of visualizations that directly support the teaching of major concepts in CS1 and CS2. These concepts include control structures, classes, interfaces, objects, inheritance, polymorphism, composition, and data structures. The integrated visualizations are intended to overcome the mismatch between what we want to teach and what most IDEs provide in the way of support for learning. The workshop will include tutorials with example programs to demonstrate how instructors can improve the learning and programming experience of their students with jGRASP. Participants are encouraged to bring programs from their own courses. jGRASP is freely available (http://www.jgrasp.org/).

All educators who teach Java will benefit from this workshop. However, it will be especially suitable for instructors who teach CS1 (introduction to programming) and CS2 (introduction to data structures and algorithms), as well as instructors for AP courses in high school. The overall objective of the workshop is to introduce faculty to the advanced pedagogical features provided by jGRASP for teaching and learning Java, including the workbench and new interactions pane. The tutorial will provide educators with an opportunity to see how each of the automatically generated software visualizations (CSD, UML, and dynamic Object Views) can be used to make learning to program a more enjoyable experience. For example, students should find the presentation (or "textbook") object views for traditional data structures (e.g., stacks, queues, lists, binary trees, etc.) particularly useful for understanding their programs. As they step through their programs in debug mode or as they invoke methods on an object that has been placed on the workbench, each presentation view is dynamically updated.

THE ACTIVE-LEARNING TRANSFORMATION: A CASE STUDY IN SOFTWARE DEVELOPMENT AND SYSTEMS SOFTWARE COURSES*

Ross Sowell, Christopher Gill, Roger D. Chamberlain, Cindy Grimm, Kenneth J. Goldman
Department of Computer Science and Engineering
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130
314 935-6160
{rsowell, cdgill, roger, cmg, kjg}@cse.wustl.edu

Mark Tranel
Public Policy Research Center
University of Missouri - St. Louis
One University Blvd. 362 SSB
St. Louis, MO 63121
mtranel@umsl.edu

ABSTRACT

We present our experience in transforming a software development course and a systems software course from a traditional, lecture-based style to an active-learning format. We outline the common changes that were made in both courses, and provide a summary of the active-learning techniques that were successfully employed. We provide quantitative and qualitative evidence that this transformation was a success. In both courses, student grades and overall satisfaction with the course were increased with the transformation to active learning, despite teaching essentially the same curriculum.

INTRODUCTION

In the last decade, research in the learning sciences has repeatedly demonstrated the benefits of active learning, when teams of students are engaged in an ongoing process of inquiry and design, centered on real-world problems structured as an extended project [8, 9, 13, 19, 20, 21]. These research findings apply to all content areas, but they are particularly compelling with regard to science, math, and engineering design [5, 10, 11]. Over and over, researchers have demonstrated that the learning environments that result in maximum retention and increased ability to transfer learning to real-world settings are very different from the lecture-plus-problem-set paradigm traditionally followed in engineering programs [2]. However, this traditional, lecture-based style persists as the dominant form of instruction in computer science.

One barrier that inhibits the widespread adoption of the active-learning style in computer science is a lack of sufficient evidence that the techniques are effective in a wide variety of computer science courses. While there have been several papers in the literature with an emphasis on active-learning, the majority of these describe general active-learning techniques [4, 14, 15, 16, 17, 18, 22] or are focused on CS 1 and CS 2 [1, 3, 6, 7, 12]. This focus is justified, as those introductory courses are taught at most institutions, but there are also many common upper-level courses that to this point have been largely ignored in the literature. Another inhibitor to adoption of the active-learning style in more courses is the time and effort that it takes for an instructor to transform his or her course. In this paper, we present further evidence that active-learning can be successful in upper-level computer science courses and provide guidelines for performing the transformation.

We present a case study of two upper-level computer science courses (one systems software course and one software development course), taught by two different instructors, that underwent a similar transformation from a traditional, lecture-based style to the active-learning format. These two courses provide an interesting comparison because both are lab-based courses in which the concepts taught are relatively straightforward and the emphasis is placed on learning how to apply them. We present quantitative evidence that this transformation was successful in the form of student grades, exam scores, and course assessments, as well as qualitative data from student interviews. We also give a summary of the key techniques employed by the instructors during the active-learning sessions to serve as a model for others that desire to transform their own course.

THE ACTIVE-LEARNING TRANSFORMATION

The Courses Pre-Transformation

"Object-oriented Software Development Laboratory" (OOSDL) focuses on practical aspects of designing, implementing and debugging object-oriented software. It is a required course for all computer science majors, and serves as a technical elective for computer engineering majors. Topics covered by this course include developing, documenting, and testing representative applications using object-oriented frameworks and C++. Design and implementation are central themes to enable the construction of reusable, extensible, efficient, and maintainable software. Prior to the spring of 2009, this

course was taught in a traditional, lecture-based style, even though all of the assignments were laboratory assignments.

"Introduction to Systems Software" (ISS) examines the process whereby computer systems manage, interpret, and execute applications. It is a required course for all computer engineering majors and serves as a technical elective for computer science majors. This course covers fundamental algorithms for numerical computation, memory organization and access, storage allocation, and the sequencing and control of peripheral devices. Prior to the fall of 2008, this course was always taught in the traditional style. The scheduled contact time was devoted to lectures, and all assignments were to be completed outside of class. These consisted of both pencil and paper exercises as well as laboratory assignments that were hands-on with the machines.

Implementation of Active-Learning

OOSDL and ISS underwent the transformation to the active-learning format in the fall of 2008 and the spring of 2009, respectively. The primary changes that were common to both courses are as follows:

- Formal comprehensive in-class lectures were largely eliminated. Instead, short, informal lecturing and reviews are now done as needed during or interspersed with the active-learning sessions.
- The traditional lectures were recorded and made available to students online for preparation prior to class, or for review afterward.
- In-class time is now spent mainly solving problems or working on programming assignments in teams, and the instructors and teaching assistants make themselves available to provide help and guidance as needed.

There are minor differences between the two courses, including how groups are formed, how in-class work is graded, and the types of work that are assigned. These are detailed below.

The OOSDL class periods are now predominantly studio sessions, in which students work in teams on assigned programming exercises that explore different programming issues and C++ language features related to each course module. The professor and teaching assistants circulate throughout the studios to answer students' questions, offer suggestions, point out issues and nuances of the exercises on which the students are working, and otherwise serve as resources for the students as they work through the exercises. The studios are largely graded for participation. Students who submit a reasonably complete set of answers earn full credit.

During the ISS class periods, students are instructed to work on the currently pending written assignment or lab assignment in small groups of their choosing. The class meets either in the regular classroom or in a computer lab as appropriate for the task that class period. The professor and teaching assistant circulate and make themselves available during the class period to help out with the work the students are doing. Whatever they do not finish in class, they are expected to finish outside of class. Assignments are typically due one week after the in-class session that they are first presented. As a result of the above, students are being graded on the active work they are

doing in class. However, since it is not due immediately, they do have the opportunity to improve it outside of class.

Examples of the exercises used for active-learning in both courses are available online:
http://www.cse.wustl.edu/~rsowell/ActiveLearningExercisesOOSDL.html
http://www.cse.wustl.edu/~rsowell/ActiveLearningExercisesISS.html

RESULTS

Exam and Project Scores

In both courses, the midterm and final exams given after the active-learning transformation were based largely on the exams from the previous semester in an attempt to measure the same things. In ISS, the same large-scale final project was assigned both semesters. In OOSDL, the project score consisted of five or six lab assignments. One less lab assignment was made after the active-learning transformation, but as a whole, the laboratory assignments covered similar material during both semesters. The median, mean, and standard deviation of these scores are provided in Table 1.

The exam scores were significantly improved with the implementation of active-learning in all the exams, except for the OOSDL final exam. In this case the scores decreased, but this decrease was not significant. We also observed a decrease in the variability of all exam scores with the implementation of active-learning. We observed a similar increase in the project scores for ISS.

				Midterm			Final			Project		
Course		Style	n	Med	Avg	Dev	Med	Avg	Dev	Med	Avg	Dev
ISS	S2006	L	24	84.8	80.8	10.8	82.0	82.4	12.5	71.0	60.5	25.8
	F2008	A	26	90.0	88.3	8.9	93.0	91.0	6.2	82.0	79.8	17.5
OOSDL	F2008	L	29	79.0	76.3	16.8	90.0	85.6	11.3	97.0	95.4	7.6
	S2009	A	30	86.0	84.1	8.7	86.0	83.8	9.4	94.0	86.9	21.4

Table 1. Exam and project scores from the courses both before and after the active-learning transformation. "L" and "A" indicate whether the course was taught in the "Lecture" or "Active-learning" style, respectively. Exam scores increased in all cases except the OOSDL final, in which no significant difference was observed, while the variability of the exam scores decreased across the board. The project scores also increased with the implementation of active-learning in ISS.

The project scores for OOSDL decreased in the spring of 2009, but this decrease was partially influenced by the scores of one student that, for personal reasons, decided to take an Incomplete in the course. If we ignore the lab scores (many of which were zeros) for this student in the computation, then the mean becomes 89.19 with a standard deviation of 16.78.

Course Evaluations

Scores from the end of semester student evaluations are shown in Table 2. Among other things, the students were asked to give a score for the following three topics:

- Overall rating of course content.
- Overall rating for teaching quality.
- Overall satisfaction with the course.

			Evaluations		
Course	Term	Style	content	teaching	satisfaction
ISS	S2006	L	0.81	0.85	0.80
	F2008	A	0.83	0.88	0.84
OOSDL	F2008	L	0.77	0.83	0.74
	S2009	A	0.76	0.76	0.76

Table 2. Evaluation scores from the courses both before and after the active-learning transformation. "L" and "A" indicate whether the course was taught in the "Lecture" or "Active-learning" style, respectively. Overall student satisfaction with the course increased in both cases.

These scores were on a Likert scale from 1 to 7, with 1 being "poor" and 7 being "excellent". The questionnaire was changed in the fall of 2008, so that the scores were on a Likert scale from 1 to 9, with 1 being "poor" and 9 being "excellent". Therefore, we have normalized the scores in Table 2.

For ISS, the mean evaluation scores increased in every category with the implementation of active-learning. For OOSDL, the scores were slightly lower for course content and teaching quality, but the overall satisfaction with the course was still higher.

One student review of OOSDL was highly negative but the others were positive, and the reviews overall indicate that while not all students viewed the approach as positive, a significant majority of the responding students viewed it highly so.

Student Interviews

An external evaluator conducted student focus groups, one each of students from the fall 2008 session of OOSDL and ISS. A total of 17 students participated in the focus groups. The focus group protocol included three topics:
Topic 1: online lectures -- strength and weaknesses
Topic 2: use of classroom time -- how different
Topic 3: use of lab time -- strengths and weaknesses of working in groups

Key themes identified in the focus groups:

- Some students commented on the lack of real-time feedback from online lectures; cannot ask questions in the course of the lecture.
- Many students found it to be advantageous that they determined when to access the lecture material and that the lecture material was available for review. They found it to be an impediment to efficient use of the on-line material not having an index to the recorded lecture. If there was some specific information that they wish to go

to for review, it was just a random process of starting and stopping the lecture to find it.
- A number of students cited technical problems with accessing the online lecture material.
- An issue frequently mentioned was the new teaching approach required more hours to complete the coursework.

Instructor Reflections

The instructor for OOSDL reported that the students worked well in groups, and though initial concerns were expressed by some students, overall their level of engagement with the material appeared high and the studio discussions were active and effective. One key suggestion made to the teams was that during a studio, the students should rotate roles so that everyone had a chance to write code, debug code, and document answers for at least some of the exercises. An unexpected but welcome development was that one of the students often did parallel development of her own solution even when she was responsible for other things, just to have the experience of solving each exercise herself.

For future offerings of the course, the instructor for ISS is considering incorporating class work that is specifically not graded in any way other than participation. This might afford students the opportunity to try more creative approaches to a problem without the fear of being penalized for something that might not work.

CONCLUSION AND FUTURE WORK

We have presented our experience in transforming a software development course and a systems software course from a traditional, lecture-based style to the active-learning format. With this transformation, student grades and overall satisfaction with the courses were increased, despite teaching the same curriculum. Finally, we presented a summary of the key active-learning techniques employed by the instructors that made the transformation a success.

While the initial active-learning transformation was certainly successful, we see several opportunities based on our evaluation for further improvement of our approach:
- The recorded lectures should be indexed and enhanced with online notes.
- Eliminate even more of the formal lecture, and replace with short, informal lecturing when necessary during or just prior to the active-learning session.
- Effectively integrate peer review into the courses.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under CPATH Grant No. CNS-0722328. Any opinions, findings, and conclusions expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Beck, L. L., Chizhik, A. W., and McElroy, A. C., Cooperative learning techniques in CS 1: design and experimental evaluation, *SIGCSE Bull*., 37, (1), 470-474, 2005.

[2] Bransford, J. D., Brown, A. L., and Cocking, R. R., editors, *How people learn: brain, mind, experience, and school*, Washington, D.C.: National Academy Press, 2000.

[3] Briggs, T., Techniques for active learning in CS courses, *J. Comput. Small Coll*., 21, (2), 156-165, 2005.

[4] Chinn, D., Martin, K., and Spencer, C, Treisman workshops and student performance in CS. *SIGCSE Bull*., 39, (1), 203-207, 2007.

[5] Edelson, D. and Reiser, B., Making authentic practices accessible to learners: design challenges and strategies, In Sawyer, R. K., editor, *Cambridge handbook of the learning sciences*, New York, NY: Cambridge University Press, 2006.

[6] Gehringer, E. F. and Miller, C. S., Student-generated active-learning exercises, *SIGCSE Bull*., 41, (1), 81-85, 2009.

[7] Gonzalez, G., A systematic approach to active and cooperative learning in CS 1 and its effects on CS 2, In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, 133-137, 2006.

[8] Hake, R., Interactive-engagement versus traditional methods: a six-thousand-student survey of mechanics test data for introductory physics courses, *Am. J. Phys*., 66, (1), 64-74, 1998.

[9] Johnson, D. R. and Smith, K., Cooperative learning returns to college: What evidence is there that it works? *Change*, (July/Aug), 26-35, 1998.

[10] Kolodner, J. L., Case-based reasoning, In Sawyer, R. K., editor, *Cambridge handbook of the learning sciences*, New York, NY: Cambridge University Press, 2006.

[11] Krajcik J. S., and Blumenfeld, P., Project-based learning, In Sawyer, R. K., editor, *Cambridge handbook of the learning sciences*, New York, NY: Cambridge University Press, 2006.

[12] Lau, K.-K., Active learning sheets for a beginner's course on reasoning about imperative programs, *SIGCSE Bull*., 39, (1), 198-202, 2007.

[13] Laws, P., Sokoloff, D., and Thornton, R., Promoting active learning using the results of physics education research, *UniServe Science News*, 1999.

[14] McConnell, J. J., Active and cooperative learning: tips and tricks (part I), *SIGCSE Bull*., 37, (2), 27-30, 2005.

[15] McConnell, J. J., Active and cooperative learning: more tips and tricks (part II), *SIGCSE Bull*., 37, (4), 34-38, 2005.

[16] McConnell, J. J., Active and cooperative learning: further tips and tricks (part III), *SIGCSE Bull*., 38, (2), 24-28, 2006.

[17] J McConnell, J. J., Active and cooperative learning: final tips and tricks (part IV), *SIGCSE Bull*., 38, (4), 25-28, 2006.

[18] Pollard, S. and Duvall, R. C., Everything I needed to know about teaching I learned in kindergarten: bringing elementary education techniques to undergraduate computer science classes, *SIGCSE Bull*., 38, (1), 224-228, 2006.

[19] Prince, M., Does active learning work? a review of the research, *Journal of Engineering Education*, 93, (2), 223-231, 2004.

[20] Redish, E., Saul, J., and Steinberg, R., On the effectiveness of active-engagement microcomputer-based laboratories, *American Journal of Physics*, 45-54, 1997.

[21] Sawyer, R. K., editor, *The Cambridge Handbook of the Learning Sciences*, New York, NY: Cambridge University Press, 2006.

[22] Schweitzer, D. and Brown, W., Interactive visualization for the active learning classroom, *SIGCSE Bull*., 39, (1), 208-212, 2007.

EXPERIENCES WITH ACTIVE LEARNING IN CS 3*

Ross Sowell, Yixin Chen, Jeremy Buhler, Sally A. Goldman, Cindy Grimm, Kenneth J. Goldman
Department of Computer Science and Engineering
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130
314 935-6160
{rsowell, chen, jbuhler, sg, cmg, kjg}@cse.wustl.edu

ABSTRACT

We report on our experiences with transforming CS 3 to an active-learning format. We have now had three separate instructors at our institution begin to integrate active learning into the course. Their approaches to integrating active learning and their experiences with it were quite different. We describe the various approaches of the instructors to the transition, provide synopses of the most effective active-learning exercises that were used, and summarize the lessons learned from these experiences. We expect that this information will be useful to anyone that desires to incorporate active learning into his or her CS 3 or similar course.

INTRODUCTION

Educational research provides strong evidence that active and collaborative learning result in a deeper and more integrated understanding of concepts, as well as significant improvement in student retention in degree programs [2, 7, 10, 16, 17, 18]. Engaged students remember concepts longer, enjoy the learning process more, and are more likely to continue. Collaborative learning builds important communication, teamwork, and leadership skills [8]. In addition, active learning in the classroom provides an opportunity to teach the creative design process through discussion and critique of student work.

In the last decade, many studies in the computer science education community have emphasized active learning. The majority of these describe general techniques [4, 11, 12,

13, 14, 15, 19] or focus on active-learning in CS 1 and CS 2 [1, 3, 5, 6, 9], while CS 3 has to this point received relatively little attention. Barriers to the widespread adoption of active-learning in CS 3 include a lack of sufficient evidence that such a transition is beneficial and for concerns of the need to cover a lot of material. Also, it takes significant time for an instructor to design and implement appropriate active-learning exercises for this type of course.

In this paper, we present our experience in transforming our CS 3 course to an active-learning format. The CS 3 course at our institution is entitled "Algorithms and Data Structures". Students study fundamental algorithms, data structures, and their effective use in a variety of applications. The course emphasizes the importance of data structure choice and implementation for obtaining the most efficient algorithm for solving a given problem. The topics covered generally include: divide-and-conquer algorithms, worst-case asymptotic analysis, sorting algorithms, decision tree lower bound technique, hashing, binary heaps, skip lists, B-trees, and basic graph algorithms. Enrollment numbers are typically between 30 and 45 students for any given session. Prior to the Spring 2008 semester, this course was always taught in the traditional lecture format.

In the past three years, we have had three different instructors teach the CS 3 course both in the traditional style and with active learning. Each instructor had a slightly different approach to the active-learning transformation and experienced varying degrees of success. We describe the various approaches, supply a collection of sample active-learning exercises that were successful, provide instructor reflections on their experiences, and summarize our lessons learned.

APPROACHES TO THE ACTIVE-LEARNING TRANSFORMATION

In the Fall 2008, Spring 2008, and Spring 2009 semesters, three different instructors that had previously taught CS 3 in the traditional, lecture-based style, taught the course again, this time incorporating active learning. Each instructor took a different approach and experienced varying degrees of success. These approaches differed in terms of the frequency and length of the active-learning sessions, whether or not lectures were recorded and assigned before class, whether or not the work was graded, and how the work was presented and discussed. The different approaches are detailed below:

- *Instructor A, Spring 2008*. In preparation for the transition to active learning, the lectures that the instructor gave in a previous offering of the course were recorded. These videos were made available to the students online, in addition to lecture notes recorded on a Tablet PC, and the textbook. In preparation for active-learning classes, the students were assigned to either watch a 15-20 minute video or complete the corresponding reading in the text. The instructor gave a short and easy quiz about the material assigned for preparation in order to learn which students had prepared. Sometimes the questions were "What is something you learned?" or "What is something that you are not clear about?" and sometimes very simple factual questions were asked. Roughly half of all the classes included some active-learning exercise. Students would divide into groups of their own choosing to work on the exercise. The instructor would circulate and help groups as questions arose. Generally, the sessions would last 30-40 minutes, with an additional 10-15

minutes of discussion afterwards. For this discussion, the instructor would ask for volunteers to share their solutions with the class, and alternate solutions were compared. Each group was asked to submit one sheet of paper with their names on it from their class work. This was graded solely on participation, which accounted for 5% of the course grade.

- *Instructor B, Fall 2008*. All class notes were recorded on a tablet PC and were posted online one or two days after class. Once every three or four classes, roughly half of the class would be devoted to an active learning session. Students were asked to divide into groups of 3-4 to work on the exercise. The instructor would circulate during the session and provide guidance and answer questions as needed. In some cases, the instructor would ask for volunteers or call on a specific group to put their solution on the board for class discussion. The work done was not graded in any way.
- *Instructor C, Spring 2009.* Half of nearly every class was devoted to an active-learning exercise. To accommodate this, students were expected to do additional reading outside of class. Roughly 15 minutes would be spent in the student-selected groups of 3-4 students each, working on the exercise. Then, two or three groups would be asked to present their solutions to the class. These would be discussed and critiqued by the other classmates as well as the instructor. The presenters would be chosen on a volunteer basis, but everyone was encouraged to present at least once during the semester. The instructor would record the names of the presenters and this would factor into a participation score that accounted for 10% of the course grade.

SAMPLE EXERCISES

There are many factors to consider when choosing an exercise for an active-learning session. If a classical problem can be presented in such a way that the students discover the algorithm or data structure on their own, then they are more likely to remember it and apply it in appropriate situations. Also, problems that have multiple solutions lend themselves well to active learning, as the presentation of alternate solutions makes students think critically about which solution they feel is preferable. Problems with multiple layers of difficulty allow for students with varying levels of understanding to make progress at their own speed and continue to be challenged.

We include a selection of successful active-learning exercises that we have used below. A complete list of exercises is available online at http://www.cse.wustl.edu/~rsowell/ActiveLearningExercisesCS3.html.

Adversary Lower Bound Technique

n Coins Problem. In this problem there are 10 coins one of which is lighter than the other 9 coins but looks the same. There is a balance scale that the algorithm is to use to determine the fake coin. The algorithm wants to minimize the number of times the scale must be used. The students are divided into groups of four. Two students in each group are to define an algorithm for the 10 coin problem. The other two students define an

adversary strategy. The two algorithm designers exchange strategies. The students are asked to prove how many weighings are optimal for 10 coins and then to try to generalize to *n* coins, and when you do not know if the fake coin is heavy or light. Based on this, it is much easier to introduce the lower bound for comparison based sorting. A benefit of this approach is it helps students learn how to write an algorithm and adversary strategy in a way that is clear and easy to understand without necessarily going to the level of code or pseudocode.

Trees and Hashing

B-Trees. After talking about secondary storage and its organization into pages, the students are asked to think about how to group the nodes in a balanced binary search tree into pages to minimize the worst case number of disk pages required in a search. This does a good job of motivating B-Trees as a generalization of binary search trees. Then, in later lectures we show how B-Tree insertion can allow you to keep the trees balanced. This also helps make it easy for the students to see the relationship between 2-3-4 trees and red-black trees.

Open Addressing Game. Students are asked to form two groups. They all leave their seats and then try to find a seat assignment using an open addressing scheme. One group uses a simple hashing function, while the other group uses a double hashing function. Each student must record how many conflicts they resolve before he or she finds a seat in which to sit. The students are then asked to discuss why one group (double hashing group) saw much fewer conflicts. This discussion went very well. Most students understood open addressing much better, and figured out the explanations for the efficiency difference between the two hashing functions, both analytically and intuitively.

Graphs

Robots. You are given a maze (an *n* x *n* grid with some edges between grid squares marked as walls that the robot cannot pass through). You are given a start location S and a goal G for the robot and asked to find the fewest steps the robot can make to get from S to G. After solving this problem the students were given the harder problem where you have two robots that start at S1 and S2. The robot starting at S1 has goal location G1 and the robot starting at S2 has goal location G2. In each time step both robots can move, however, they cannot share the same location. Find the minimum time steps needed for both robots to reach their goals (and also find the corresponding solution). This second version is much harder since it requires them to change the state space so that each vertex in the graph corresponds to a pair of robot locations. This worked out very well. Even the students who did not figure out how to solve the second version benefited when we went over the solution at the beginning of the next class.

Savage and Human Game. Students were asked to play a river-crossing flash game "savage and human". Then they were asked to formulate it into a graph theory problem. It went very well as most students found it very interesting and figured out that it amounts to finding a shortest path in the state space graph. They can start to see that many real-world planning and scheduling problems can be translated into a graph theory problem.

Dijkstra's Algorithm. After we discussed breadth-first search (BFS) and proved that it can find the shortest path for graphs where each edge has a unit weight, the students were asked to generalize the idea to graphs with positive weights. After being given the hint of breaking each edge with weight k to k edges with unit costs, some students were able to derive Dijkstra's algorithm and easily prove its correctness, using the BFS result as a lemma. It was a satisfying experience for students who can figure it out, and it helps students gain a deeper understanding of Dijkstra's algorithm.

INSTRUCTOR REFLECTIONS

- *Instructor A, Spring 2008*. The instructor had a generally positive experience with the active-learning transformation. Many students appreciated being able to re-watch portions of the lecture at their own convenience. One item that she felt was particularly important was to try to incorporate different levels of difficulty into each exercise. For slower groups, they at least have the satisfaction of understanding the first level or two, while more advanced groups are still challenged with the later levels. When circulating the class, she thought it was important to visit the groups that did not seem to be doing anything first. Sometimes they needed a little help to get started. She also noted that it was important to carefully consider what material to include for active learning. For example, anything that the student has seen previously, is far less interesting for active learning. Finally, she felt that collecting sheets of paper from each group was helpful, as browsing through them gave her some idea of the thought process of some of the groups that she did not get to visit.
- *Instructor B, Fall 2008*. The instructor had a largely negative initial experience with the active-learning transition. Whether it was the content or difficulty or organization, his students did not seem to get a lot out of it, even though they would do the work. Some students did not seem to have an opinion one way or the other, but others actually resented it. They did not understand why they were doing more exercises similar to their homework in class. The instructor is not sure if he chose the best exercises to try active learning with, and intends to make another attempt at active learning in the future.
- *Instructor C, Spring 2009*. The instructor observed that students were more engaged, more interested, and generally got more out of the class when they participated in the discussion. The students also exercise independent, creative thinking, rather than just learning what is in the text. It also serves as an advantage for the instructor, since he gets the opportunity to better know the students. The drawbacks include that sometimes the discussion veers off topic or a group presents a very strange algorithm. In these cases, it is the instructor's responsibility to steer the discussion back on track. He also noted that he is unable to cover the same amount of material in class, so some things have to be left to the reading. Students must do more reading to cover the same material. He feels that the students who participate learn more, but not everyone wants to participate. The main issue is how to adapt to all types of students.

CONCLUSION AND FUTURE WORK

We have presented the experiences of three instructors that each incorporated active learning into CS 3. We have described the various approaches and techniques used, provided a set of exercises that were successful, and presented the reflections of the instructors on their experiences. Our experiences with active learning in CS 3 have yielded successful techniques that are now recommended to others as well as posed challenges for future investigation.

Successful techniques include:

- Recording the traditional lectures. Students appreciate being able to watch the lecture or read the text to prepare for class. Having the lectures indexed allows the student to watch only the portions that he or she needs. Being able to re-watch difficult portions of the lecture is another asset.
- Brief, easy quizzes at the beginning of class can help encourage student preparation for active learning and alert the instructor to students that are not prepared.
- Exercises with multiple solutions and multiple levels of depth work best for active-learning sessions.

Challenges include:

- How to select the student groups? Letting students choose their own groups or assigning them based on seating may not be the best option. Students have different levels of experience, and it may be best to group them according to their level of experience and ability.
- How to effectively integrate peer review? So far, most of the critique has come from the instructor or from other students in a class discussion. It would be nice to have one group critique the algorithm, data structure, or proof of another group.
- Most of the lecturing is done at the beginning of class, followed by the active-learning session. It may be more effective to lecture briefly, and then begin the active-learning session. The session could then be interspersed with brief, informal lectures throughout.

ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under CPATH Grant No. CNS-0722328. Any opinions, findings, and conclusions expressed are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Beck, L. L., Chizhik, A. W., and McElroy, A. C., Cooperative learning techniques in CS 1: design and experimental evaluation, *SIGCSE Bull*., 37, (1), 470-474, 2005.

[2] Bransford, J. D., Brown, A. L., and Cocking, R. R., editors, *How people learn: brain, mind, experience, and school*, Washington, D.C.: National Academy Press, 2000.

[3] Briggs, T., Techniques for active learning in CS courses, *J. Comput. Small Coll.*, 21, (2), 156-165, 2005.

[4] Chinn, D., Martin, K., and Spencer, C, Treisman workshops and student performance in CS. *SIGCSE Bull.*, 39, (1), 203-207, 2007.

[5] Gehringer, E. F. and Miller, C. S., Student-generated active-learning exercises, *SIGCSE Bull.*, 41, (1), 81-85, 2009.

[6] Gonzalez, G., A systematic approach to active and cooperative learning in CS 1 and its effects on CS 2, In *SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education*, 133-137, 2006.

[7] Hake, R., Interactive-engagement versus traditional methods: a six-thousand-student survey of mechanics test data for introductory physics courses, *Am. J. Phys.*, 66, (1), 64-74, 1998.

[8] Johnson, D. R. and Smith, K., Cooperative learning returns to college: What evidence is there that it works? *Change*, (July/Aug), 26-35, 1998.

[9] Lau, K.-K., Active learning sheets for a beginner's course on reasoning about imperative programs, *SIGCSE Bull.*, 39, (1), 198-202, 2007.

[10] Laws, P., Sokoloff, D., and Thornton, R., Promoting active learning using the results of physics education research, *UniServe Science News*, 1999.

[11] McConnell, J. J., Active and cooperative learning: tips and tricks (part I), *SIGCSE Bull.*, 37, (2), 27-30, 2005.

[12] McConnell, J. J., Active and cooperative learning: more tips and tricks (part II), *SIGCSE Bull.*, 37, (4), 34-38, 2005.

[13] McConnell, J. J., Active and cooperative learning: further tips and tricks (part III), *SIGCSE Bull.*, 38, (2), 24-28, 2006.

[14] J McConnell, J. J., Active and cooperative learning: final tips and tricks (part IV), *SIGCSE Bull.*, 38, (4), 25-28, 2006.

[15] Pollard, S. and Duvall, R. C., Everything I needed to know about teaching I learned in kindergarten: bringing elementary education techniques to undergraduate computer science classes, *SIGCSE Bull.*, 38, (1), 224-228, 2006.

[16] Prince, M., Does active learning work? a review of the research, *Journal of Engineering Education*, 93, (2), 223-231, 2004.

[17] Redish, E., Saul, J., and Steinberg, R., On the effectiveness of active-engagement microcomputer-based laboratories, *American Journal of Physics*, 45-54, 1997.

[18] Sawyer, R. K., editor, *The Cambridge Handbook of the Learning Sciences*, New York, NY: Cambridge University Press, 2006.

[19] Schweitzer, D. and Brown, W., Interactive visualization for the active learning classroom, *SIGCSE Bull.*, 39, (1), 208-212, 2007.

SOMETIMES STYLE REALLY DOES MATTER*

David Reed
Department of Computer Science
Creighton University
davereed@creighton.edu

ABSTRACT

Programming, like any creative endeavor, involves some personal style choices on the part of the programmer. Within the computer science education community, there are some programming style conventions that have been widely agreed upon, because following them unequivocally leads to programs that are easier to read and maintain. In other cases, a programmer might reasonably choose between competing styles, each of which provides similar advantages. This paper describes three instances where programming style is more than just aesthetics - following these conventions can actually help a beginning programmer to avoid mistakes and better understand the underlying programming concepts that they are utilizing.

1 INTRODUCTION

Programming is a fascinating blend of engineering and art. While software engineering provides tools and methodologies for building reliable and cost-effective software systems, the creative aspect of programming and the opportunities for personal expression in the programming process cannot be denied. In his 1974 Turing Award acceptance speech [1], Donald Knuth wrote:

> *"When I speak about computer programming as an art, I am thinking primarily of it as an art form, in an aesthetic sense. The chief goal of my work as an educator and author is to help people learn how to write beautiful programs. ... Programmers who subconsciously view themselves as artists will enjoy what they do and will do it better. "*

In both art and programming, style is not a purely subjective matter. For example, there are some foundational elements of painting (i.e., brush techniques, effective use of color) that must be mastered by the budding artist and integrated into his or her aesthetic.

Similarly, within the computer science education community there are some programming style conventions that have been widely agreed upon. For example, any introductory text on programming will mention the importance of selecting meaningful names for variables, since names that suggest their purpose make code easier to read and maintain.

In addition to universally accepted conventions, there are some instances where different programming style aesthetics compete, but without a clear advantage. For example, some programmers still argue religiously over where to put the opening curly-brace on an if statement or loop: some advocate placing the curly-brace at the end of the opening line for that statement/loop, while others advocate placing it alone on the next line. The debate persists because there is no clear advantage to either style. Proper indentation ensures that the code is easy to read, regardless of where the opening curly-brace to placed.

The following sections describe three style conventions whose benefits go beyond aesthetics. Each convention can assist the beginning programmer in avoiding errors and in developing a stronger understanding of the underlying programming concepts involved. These three conventions are specific to the Java language, although the general concepts can be applied to other object-oriented languages.

CATEGORY	NAMING CONVENTION
classes interfaces	Should be nouns, in mixed case, with the first letter uppercase, e.g., `CustomerDatabase`.
methods	Should be verbs, in mixed case, with the first letter lowercase, e.g., `getCustomerID`.
instance/class variables parameters local variables	Should be nouns, in mixed case, with the first letter lowercase, e.g., `numberOfEntries`. Variable names should not start with '_' or '$' characters, even though both are technically allowed.
class constants	Should be all uppercase with words separated by underscores, e.g., `DEFAULT_SIZE.`

Figure 1. Java naming conventions (from http://java.sun.com/docs/codeconv/).

2 JAVA NAMING CONVENTIONS

The first style convention that can be beneficial to beginning programmers is a non-controversial one: following consistent naming conventions. Naming conventions, guidelines for choosing the names of program elements, have been used extensively within the software industry to develop uniform code that is easier to read, maintain, and integrate across projects. Even in the smaller-scale programs typically encountered by beginning programmers, naming conventions can clarify the roles of different program elements and can provide guidance when making design choices. For example, if a student knows that constants are represented using all-uppercase names, then he can more easily identify them when scanning code. Similarly, following the convention that object and class names are nouns whereas method names are verbs provides design guidance to beginners, making name choices less arbitrary and leading to code that reads more

naturally. Interestingly, research described in [2] suggests that poorly chosen identifier names can not only hinder comprehension, but may also indicate confusion on the part of the programmer and potential bugs in the code.

While all programming languages support various naming conventions, Java is noteworthy for its use of a central, well-established set of naming conventions that is published by Sun along with other references (Figure 1). Since all official Java code, including the standard libraries, follows these conventions, a student reviewing code can rely upon the conventions to help her understand the code. For example, the following Java expressions, although similar syntactically, suggest very different meanings:

```
Foo.bar()                    foo.bar()
```

By knowing the Java naming conventions, it is possible to distinguish between these at first glance: the first represents a call to the static method bar in the `Foo` class, while the second represents an instance method call on an object named `foo`. A beginning programmer who learns these naming conventions and follows them in his own code will similarly experience the benefits of readability and will be able to seamlessly integrate his code with existing programs.

The value of following the Java coding standards is demonstrated by a survey of introductory Java texts. Some texts (e.g., [3,5,7]) explicitly refer to the Java standards when describing the naming conventions used in the text. And while others (e.g., [4,8]) may not explicitly refer to the Java standards, the conventions used throughout the texts match the Sun coding conventions. Despite the clear message being sent by textbook authors, however, it is clear that many students are not being taught the importance of following the Java naming conventions. A search of online introductory Java courses turns up numerous violations, most notably nouns for method names and inconsistent capitalization.

3 CONSISTENT USE OF "this."

The next style convention, the consistent use of the "this." prefix for internal method calls and instance variable references, is more controversial but has great potential for assisting beginning programmers. Recall that, in Java, a dot is used to denote object ownership when calling a method or accessing an instance variable of an external object. For example, `die6.roll()` specifies calling the `roll` method that *belongs to* (and is being applied to) an object named `die6`. Similarly, `Math.PI` references the public class variable `PI` that *belongs to* the predefined `Math` class. In cases such as these, the prefix before the method call or variable reference is required in order to identify the particular object being acted upon. In contrast, internal method calls and instance/class variable references do not require prefixes, as the current object is the implicit target of any actions/references. For example, another method within `die6`'s class definition could call the `roll` method as simply `roll()`. Likewise, a method of the `Math` class can access the `PI` class variable as simply `PI`.

In my 10+ years of teaching intro Java programming, I have found that students are often confused by the inconsistent structure of external and internal method calls and variable references. As soon as they begin to appreciate that the dot notation denotes ownership and the application of a method to a particular object, they encounter internal

method calls where the object is not explicit. Similar confusion arises when distinguishing between local variables and instance variables within a class. Since there is no dot prefix to identify instance variables as belonging to the object, references to instance variables within methods have the exact same form as accesses to local variables.

While Java does not require explicit prefixes on internal method calls and internal references to instance/class variables, it does allow for them using the "this." prefix. The Java keyword "this" denotes the current object, so adding it to the prefix of internal method calls and instance/class variable references does not change their meanings. It does, however, make the format of internal calls and references consistent with their external counterparts. For example, consider the implementation of the `Die` class shown in Figure 2. The class has a single instance variable, `numSides`, which stores the number of die sides (six by default). Whenever, that instance variable is referenced in the class, it is referenced as `this.numSides`, highlighting the fact that it belongs to the current object. Similarly, the internal method call `this.getNumberOfSides()` uses the "this." prefix to highlight that the method is being applied to the current object.

The cost of making "this." an explicit prefix on instance variables and method calls is a minor one - the additional length of the references. The benefits, however, are major. First, the consistent use of "this." makes object ownership explicit and consistent. Every method call, whether it be internal (applied to the same object) or external (applied to another object) has the same format: `OBJECT.METHOD()`. Similarly, every reference to an instance variable explicitly lists the object: `OBJECT.VARIABLE`. This consistent syntax makes object ownership more transparent to a beginning programmer and reinforces an object-oriented mindset. Second, the consistent use of the "this." prefix makes the distinction between instance variables and parameters/locals clear. When the student reviews a class definition and sees the "this." prefix on a variable, he knows that this is an instance variable that belongs to the object (and persists between method calls). A variable without the prefix is a parameter or local variable that belongs to a method and exists only while that method is executing. This explicit distinction makes it easier for a beginner to understand object state and variable scope. Third, the use of the "this." prefix for instance variables avoids naming conflicts that might otherwise arise. For example, in the Die class (Figure 2) there is an instance variable named `numSides` and a constructor parameter with the same name. Without the "this." prefix, different names would have to be chosen for one of the variables, which usually leads to an abbreviated and less meaningful name such as `sides`.

Despite the advantages of the explicit "this." prefix, no introductory texts consistently use this style. Most refer to "this" in a more limited context, especially with respect to avoiding naming conflicts (e.g., [8,9,10]). The strongest support for this convention comes in [5], where Horstmann encourages programmers to try out the style of explicitly listing the "this." prefix. In personal conversations, Horstmann has expressed a personal preference for this style, but admits to not including it in his text because it is not widely used. Anecdotally, I have observed less confusion and a deeper understanding of object concepts among my introductory-level students since I began emphasizing "this."

```
public class Die {
  private int numSides;

  /** Constructs a 6-sided die. */
    public Die() {
        this.numSides = 6;
    }

  /** Constructs an arbitrary die.
   *    @param numSides the number of die sides */
    public Die(int numSides) {
        this.numSides = numSides;
    }

  /** Accessor method for the number of die side.
   *    @return the number of sides */
    public int getNumberOfSides() {
        return this.numSides;
    }

  /** Simulates a roll of the die.
   *    @return a random int between 1 and the number of sides */
    public int roll() {
        int currentRoll = (int)(Math.random() *
       this.getNumberOfSides()) + 1;
        return currentRoll;
    }
}
```

Figure 2. A class for modeling dice (and demonstrating the use of "this.").

4 MAIN METHOD FOR CLASS TESTING

Over the past decade, studies have shown that integrating unit testing into introductory computer science courses can help students learn the process of designing and testing software (see [11,12], for example). While many Java IDEs, integrate unit testing functionality, unit testing has still not achieved widespread use at the introductory level in colleges. A justification commonly given by instructors is that introductory programming courses are already too full of syntactic and technical details - adding another programming tool and set of techniques is just not feasible.

Some of the benefits of unit testing can be achieved using pedagogically inspired IDEs such as BlueJ and Dr. Java. Using BlueJ, a student can call any method on an object by right-clicking on the object icon, selecting the method, and entering parameter values in text boxes. This direct manipulation of objects is very intuitive for beginning students, allowing them to test each method individually (and alleviating the need for "public static void main" altogether). BlueJ and Dr. Java also have interaction panes, where the student can enter snippets of code and test class methods without the overhead of a separate driver class (or unit test). While these features are handy, they do not replace the full functionality of unit testing. The student must manually test each method of each class, and there is no automatic process for retesting code after making modifications.

```
public class Die {
  private int numSides;

  /** Constructs a 6-sided die. */
    public Die() {
        this.numSides = 6;
    }

  /** Constructs an arbitrary die.
   *    @param numSides the number of die sides */
    public Die(int numSides) {
        this.numSides = numSides;
    }

  /** Accessor method for the number of die side.
   *    @return the number of sides */
    public int getNumberOfSides() {
        return this.numSides;
    }

  /** Simulates a roll of the die.
   *    @return a random int between 1 and the number of sides */
    public int roll() {
        int currentRoll =
(int)(Math.random()*this.getNumberOfSides()) + 1;
        return currentRoll;
    }

  /** Main driver method for testing Die objects.
   *    Constructs a 6-sided die and displays 20 rolls, then
   *    constructs a 4-sided die and displays 20 rolls.      */
    public static void main(String[] args) {
        Die d6 = new Die();
        System.out.print(d6.getNumberOfSides() + "-sided die: ");
        for (int i = 0; i < 20; i++) {
            System.out.print(d6.roll() + " ");
        }
        System.out.println();

        Die d4 = new Die(4);
        System.out.print(d4.getNumberOfSides() + "-sided die: ");
        for (int i = 0; i < 20; i++) {
            System.out.print(d4.roll() + " ");
        }
        System.out.println();
    }
}
```

Figure 3. Die class with a main method for testing.

In most programming texts, classes are tested using a separate "driver" class. The driver contains a main method for creating an object of the class and calling methods on that object. Thus, to test the class, the programmer must add the driver to the project and separately compile and execute that driver. Unfortunately, requiring one or more drivers for every class in a project can lead to a very cluttered project space. An alternative style that avoids project clutter while achieving some of the advantages of unit testing involves integrating the driver's main method directly into the class itself. For example, Figure 3

shows the `Die` class with a main method added. This method creates two different die objects and displays the results of method calls on those objects.

By embedding the main method in the class itself, the programmer obtains the advantages of the driver class without the clutter of additional classes in the project. Popular IDEs such as netBeans and Eclipse allow the programmer to separately compile files in the project and execute any file that has a main method. Thus, by embedding the testing code inside a main method in the class, the programmer achieves some of the benefits of unit testing without requiring any additional tools. Each class can be tested independently (by executing its main method), and it is straightforward to retest a class if any related code is updated.

While this style of class design/testing does avoid cluttering the project with driver classes, one drawback is that it clutters individual classes with main methods. For example, the `Die` class from Figure 2 models a real-world die object, capturing its state and behavior. Adding a main method for testing purposes (Figure 3) is not part of the software model, and seeing it in the class definition or accompanying javadoc documentation for the class might confuse some beginning programmers.

5 CONCLUSION

Sometimes, programming style is more than just aesthetics. The consistent use of pedagogically sound conventions can actually help the beginning programmer to avoid mistakes and better understand underlying concepts. For example, following the Java naming conventions can lead to code that is easier to read and maintain. Consistently using the "this." prefix for instance/class variables and internal method calls can help a student understand object-oriented concepts and more easily differentiate between variable types. Adding a main method to every class in a project can achieve some of the benefits of unit testing without introducing any new concepts or tools. Conventions such as these can not only assist the budding programmer in writing "beautiful" programs (as Knuth would put it), but can also help them to understand difficult object-oriented and software engineering concepts.

REFERENCES

[1] Knuth, Donald. Computer Programming as an Art, *Communications of the ACM*, 17(12), 1974.

[2] Butler, Simon. The Effect of Identifier Naming on Source Code Readability and Quality. In *Proceedings of the Doctoral Symposium for ESEC/FSE (Amsterdam)*, ACM, 2009.

[3] Dale, Nell and Weems, Chip. *Programming and Problem Solving using Java, 2nd edition*, Jones and Bartlett, 2007.

[4] Lewis, John and Loftus, William. *Java Software Solutions, 6th edition*, Addison-Wesley, 2009.

[5] Horstmann, Cay. *Big Java, 3rd edition*, John Wiley and Sons, 2008.

[6] Sun Developer Network. *Code Conventions for the Java Programming Language*, 1999. Accessed online at http://java.sun.com/docs/codeconv/.

[7] Morelli, Ralph and Walde, Ralph. *Java, Java, Java, Object -Oriented Problem Solving, 3rd edition*, Prentice Hall, 2006.

[8] Liang, Daniel Y. *Introduction to Java Programming - Comprehensive Version, 6th edition*, Prentice Hall, 2007.

[9] Deitel, Harvey and Deitel, Paul. *Java How to Program, 7th edition*, Prentice Hall, 2007.

[10] Wu, C. Thomas. *A Comprehensive Introduction to Object-Oriented Programming with Java*, McGraw-Hill, 2008.

[11] Olan, Michael. Unit Testing: Test Early, Test Often, *Journal of Computing Sciences in Colleges*, 19(2), 2003.

[12] Desai, Chetal, Janzen, David S. and Clements, John. Implications of Integrating Test-driven Development into CS1/CS2 Curricula. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education, ACM SIGCSE Bulletin*, 41(1), 2009.

ONLINE COMMUNITIES IN THE ERA OF THE INFORMATION REVOLUTION*

Igor Balsim, Elie Feder , and Sarwar Jahangir
Department of Mathematics and Computer Science
Department of biology
Kingsborough Community College of the City University of New York
2001 Oriental Blvd, Brooklyn, NY 11235
(917)521-1675
ibalsim@gmail.com

ABSTRACT

This paper expounds upon the innovative use of social software to build and organize an environment for healthy and thriving online communities that are focused on interdisciplinary research and education. The exposition focuses on a community engagement model that focuses on peer-to-peer support, learning, sharing, and networking, to allow users to help each other in solving problems. Examples of online tools and successful online communities are provided to illustrate the various possibilities.

1. INTRODUCTION: THE INDUSTRIAL REVOLUTION AND THE INFORMATION REVOLUTION

When studying the history of the Industrial Revolution and comparing it with the Information Revolution, one finds many parallels. The onset of the Industrial Revolution marked a major turning point in human history. Major changes in agriculture, manufacturing, mining, and transport had a profound effect on the socioeconomic and cultural conditions of the world. This revolution was one of the most important events in history because almost every aspect of daily life was eventually influenced in some manner. The period of the revolution is associated with innovations that helped energize the new socioeconomic changes [8, 10]. Knowledge of the innovations was transferred through various means including: (i) networks of informal philosophical societies whose members met to discuss science and its application to manufacturing; (ii) publications,

such as encyclopedias that contained vast amounts of information regarding the science and technology of the first half of the Industrial Revolution; (iii) study tours [1, 4]. The Industrial Revolution can be characterized by the systematic application of science and empirical knowledge to the production process [3].

The Information Revolution constitutes a historical epoch on a par with the Industrial Revolution. In the second half of the twentieth century, the world has evolved as a result of the Information Revolution and its influence on current socio-economic and technological trends [7]. The development of technologies such as computers, digital communications, and microchips, has led to dramatic reduction in the cost of obtaining, processing, storing, and transmitting information. The massive explosion of information has been facilitated by the technological revolution in computer applications and telecommunication networks. The increase in the development of Information and Communication Technologies has revolutionized various sectors, including manufacturing, business, science and technology, schools and homes [15]. The current age marks a new age of transformations and profound economic, social, and political possibilities. Fukuyama argued that the wealth of information at consumers' fingertips-combined with the ability to seek out and shop for exactly what they want without regard to borders and other restrictions-would lead to a market-led democratization [5]. At the base of any information system of any scale is a communications system. With the rise of the Internet, we now have the means to meet the most demanding communications requirements [2].

2. THE USE OF WEB 2.0 AND ONLINE COMMUNITIES IN THE INFORMATION REVOLUTION

We are presently entering yet another, more progressive phase of the Information Revolution due to the development of Web 2.0, an interactive collaboration of online social communities. This new juncture is distinguished by the forces of sharing, peering, and networking that directly link more than a billion people and create a real world interface. It represents the most robust platform of new networks of people, knowledge, objects, devices, and intelligent agents, where innovations and social trends spread with viral intensity. Social software can be used to create an environment that provides a sustainable architecture of participation. The Web 2.0 provides an opportunity to support multiple conversations at varying levels of engagement, reaching both a target audience and their extended social network [17]. This suggests the innovative use of social software to build and organize an environment for healthy and thriving online communities that are focused on interdisciplinary research and education. We recommend communities which focus on peer-to-peer support, learning, sharing, and networking, to allow users to help each other in solving problems. Some of the objectives of such learning communities are to provide education and expertise for their members, to establish a reputation as a successful tool of education, and to give visitors a place to inquire and learn to improve their educational and research careers.

Communities of students and faculty can be used to create and support a platform for improvement of communication, collaboration, and innovation in the development of ideas. These communities can foster a wealth in learning and scientific discovery by creating goals that are focused on the people in the community [17]. The greatest support

for the development and growth of online learning communities is an environment where collaboration, knowledge, and innovations thrive via support and security.

Peer collaboration of online communities is built through volunteers that are enthusiastic and motivated to offer their time, knowledge, and skills to help support the communities. Such volunteers gain experience, exposure, and connections. They earn respect and status within the community that is highly valuable in their future careers. The necessary conditions for the success of the online communities are that each community is based on the development and transfer of information that can be bifurcated into small parts, with low integration costs. The community demands a leader who can help guide and manage its social interactions by setting up a protocol for cooperation, and motivating and coordinating collective actions over long periods of time. The rationale for the success of such communities is that new economics via technology have low costs and higher benefits of producing information. The online communities are often more efficient than firms or markets at properly allocating time and attention. This is made possible through attracting a more diverse talent pool of participants that enjoy the experience of peer production [17].

The wide accessibility, adaptability and expeditious utilization of information from online communities promote the innovation power of a community by tapping into good ideas that would otherwise be hard to find. Open source makes the inner workings of technology apparent, allowing others to build upon and improve novel innovations [20]. An additional benefit of online communities is that the software technology it makes use of can also provide the ability to measure and quantify the success of the participants of the network.

3. VARIOUS TECHNOLOGIES AND TOOLS

We describe some of the various technologies that are used to build the social communities [9].

Blogs are written expressions of ideas that encourage commentary regarding the ideas presented. They combine text, images, links to other blogs, web pages, and other media related to its topic. Large networks of blogs create a *blogosphere* that consists of relationships between blogs and their authors [9].

Microblogs are a form of multimedia blog that allows users to write brief messages, text updates or micromedia and to publish them. These messages can be made public or restricted to a group which is chosen by the user. They can be submitted by a variety of means, including text messaging, instant messaging, E-mail, digital-audio, or the web [19].

Podcasts are an audio and video expression of ideas that encourage comments [9].

Social Networks such as LinkedIn, MySpace, or Facebook allow members to maintain profiles, connect with each other, and interact [9].

Wikis are an informational tool which allows for shared contribution and responsibility for maintaining information. The community screens all possible changes based on its rules and regulations. Wikis are useful as a tool for collaboration in group projects [9].

RSS (Really Simple Syndication) is a tool that brings updates from blogs, news, and various websites [9].

Twitter is a free social networking and micro-blogging service where users have up to 140 characters to communicate to others. Users can link to posts that can be made public or kept private. Twitter is utilized in various settings as an invaluable tool towards meeting difficult objectives [11, 18, and 19]. Many libraries use Twitter as an educational and professional development tool, as well as an advertising tool. Teachers use Twitter to both network and *tweet* with more experienced teachers and coaches. Twitter provides a direct source of professional support and an online learning community for new teachers who lack mentoring support and coaching in their school districts. Additionally, Twitter can be a very useful resource for supporting research communities. It provides users instant access to experts in each field of interest [16].

4. EXAMPLES OF ONLINE COMMUNITIES

In this section, we give examples of successful online communities which can be subdivided into the following four categories: Social, Business, Education, and Research. These examples serve to illustrate the applicability and usefulness of online communities in all walks of life.

4.1 Social

CarePages is an online communication system designed to help patients cope with their illnesses and stay in touch with family and friends. It is a specialized blogging system designed for patients to post updates for family and friends who can, in turn, send support messages for the patients. Some benefits of this community are that it can lessen the burden on patients and their families, and that it allows well-wishers (both friends and strangers) to post friendly messages [9].

American Cancer Society Cancer Survivors Network is a vibrant source of support for cancer victims. The network lets patients help each other by sharing personal stories and life experiences. Such networks can be used as referral systems for recommending high quality physicians based upon personal experiences [17].

4.2 Business

InnoCentive consists of 90,000 registered scientists from 175 countries to provide solutions to companies in science and technology. It links experts to unsolved research and development problems by tapping the talents of a global science community without having to employ all of its workers on a full time basis. Werner Mueller is a chemist who was able to solve a given problem on InnoCentive. A pharmaceutical company required a cost effective method of producing an early stage raw material. Mueller recognized this product from decades of experience as a chemist, submitted a solution and received $25,000 [17].

Linus Torvalds created a simple version of the *Unix* operating system and shared it with other programmers via an online bulletin board. This creation eventually led to the

creation of *Linux*, a multibillion dollar ecosystem operating system that is useful for hosting Web servers, databases, and supercomputers. Linux is under general public license, i.e., anyone could use it for free, provided that they make changes that are available to others. An informal organization emerged to manage the ongoing development that gets input from thousands of volunteer programmers. Many companies consider Linux to be keystone software. IBM donated large volumes of proprietary software code and established teams to build Linux open source communities [17].

4.3 Education

One important facilitator of education is the ability to access relevant information in an efficient and accurate manner. In past generations, multivolume encyclopedias served this purpose. These are often difficult to access and utilize to find desired information. Today, *Wikipedia* is the largest encyclopedia in the world. It is created by volunteers who are passionate about their work, runs on an open platform and builds on the Web software wiki. It contains four million articles in 200 languages and provides free access to the sum of all human knowledge. Collaboration among users improves its content over time. Due to its extreme openness and its allowance of anybody to be an editor to most articles, it is vulnerable to inaccuracies and false statements. Despite this weakness, it still has a high accuracy evaluation report. *Nature* magazine performed a comparative analysis between Wikipedia and *Britannica*. Wikipedia contained four inaccuracies for every three from Britannica [6]. It is a dynamic and evolving body of knowledge that is adaptable to fix its mistakes. It is continuously growing, adding new entries, reviewing, and updating new facts. As a result, it taps an infinite wealth of talent, energy, and insight that far exceeds Britannica [17].

Wikipedia has started wiki sister projects such as *Wikiversity*, a project that is devoted to creating learning resources, learning projects, and research for use in all levels, types, and styles of education from pre-school to university, including professional training and informal learning. This is in line with the Web 2.0 philosophy of advancing knowledge and education via new technology. Similarly, by November 2007, MIT completed the initial publication of virtually the entire curriculum, over 1,800 courses in 33 academic disciplines, in there online project known as *Open Course Ware (OCW)* [17]. The MIT faculties are passionate about their teaching and are keen to see their work benefit global society [12].

The educational process can be greatly enhanced through the development of online learning communities. A learning community is a network of people who care about a specific issue, problem, or debate. This notion can be expanded to include a wide range of experts in many different fields via the interactive learning process that develops as the community works together. The members of each learning community can pose challenging questions to one another, point to valuable resources, and provide instant responses to questions posed. The use of online learning communities can support students' learning process with collaborative assignments, facilitation of active discussion, and promotion of the development of critical thinking and research skills. Learning with technology allows participants to explore issues with wide applications at there fingertips. Students participating regularly in an online course will improve their ability to use technology with confidence. The students play a much more active role in their learning,

while the teacher becomes a mentor and a guide. In such an environment, collaboration between student-student and teacher-student are essential to success. The outcomes and benefits of an online learning community are due to a great wealth of knowledge acquisition. Students learn about the technology through its use. They learn about themselves and their own learning styles, and about how to collaborate with others in the problem solving process. They become increasingly confident in their abilities, feel empowered to work in a manner that best suits them, and seek out the information they need for a given task [13]. They are forced to learn how to think quickly, to adapt to changing conditions, to build alliances to address large-scale challenges, and to work comfortably in a global information environment. These skills are the keys for the creation and communication of ideas. They provide a way to develop an educational system that can help students learn to work in a world culture [14].

We give some examples of the learning projects that have been used in various learning communities. Electronic field trips are used to help students join teams of researchers, scientists, and technicians exploring distant regions, such as a rain forest, Antarctica, or Mars. Telerobotics tools combine research and communication, making it possible for students to direct a telescope to look out into space from their classroom. In globe projects, students from around the world collect, compare, analyze data on air temperature, air pressure, and wind speed. With the use of online science centers, visualization tools make it possible for students to display whatever data they collect and wish to share. These are but some examples of how online communities can be a powerful tool in the educational experience [14].

4.4 Research

The *Human Genome (HG) Project* is the key to eliminating dreadful diseases such as diabetes and cancer. Its applications in agriculture and ecology could help end hunger. Pharmaceutical firms abandoned proprietary HG projects to support open collaborations that will cut costs, accelerate innovation, create more wealth, and help society reap benefits of HG research more quickly. A robust scientific commons is the best way to ensure the full potential of the genomic revolution [17].

Open Wet Ware is an MIT online research community designed to promote the sharing of information, know-how, and wisdom among researchers and groups who are working in biology and biological engineering. It has twenty labs at different institutions around the world that use its website to swap data, standardize research protocols, share materials and equipment. It provides more dynamic ways for publishing and evaluating scientific work. It is built on the *MediaWiki* software that provides blogs, protocols-share techniques, and discussion groups [17].

The *Google* enterprise has grown to be a leader in both computer hardware and software due to social networks and relations, without any media or promotions. People feel emotionally engaged to its search engine. Its founders, Sergey Brin and Larry Page, provide strong leadership with grand ambitions that are the key factors of Google's success. They motivate a variety of scientists, mathematicians, and engineers by engaging them with challenging problems and creating an environment that solicits new ideas and constant growth and progress. Thus, the Google Company is constantly introducing new

features and services at a very rapid pace. It is working on research projects, such as molecular biology and genetics, by creating search engines for massive scientific data using data bases and computing power. Such projects may lead to significant breakthroughs in science, medicine, and health. Google is also in the process of digitizing millions of books from various libraries with the objective of using search engines on all the acquired information. Thus, it promises to create a far reaching educational, scientific research, and social impact [21].

5. CONCLUSION

This paper described the force of collaboration through the online Web 2.0 environment. It is a new way for social communities to learn, innovate, discover and interact by providing motivation, support, and experience. The key factor for success in such communities is harnessing external knowledge, resources and capabilities [17]. We illustrated with examples successful communities from the various fields of social, business, education and research. The availability of free access to the massive information due to such communities is unprecedented throughout the history of the world.

REFERENCES

[1] Beck B.R., *World History: Patterns of Interaction*. Evanston, Illinois: McDougal Littell, 1999.

[2] Buddenberg, R.A., *The Fragmented and Incomplete Revolution in Military Affairs*, http://web.nps.navy.mil/~budden/lecture.notes/ind_rev.html, retrieved December 13, 2009.

[3] Deane, P., *The Industrial Revolution*, Cambridge, UK: Cambridge University Press, 1965.

[4] http://encyclopedia.stateuniversity.com/pages/10562/Industrial-Revolution.html#ixzz0Zb7mFJQF

[5] Fukuyama, F., The Great Disruption, *The Atlantic Monthly* 283 (5), 55-80, May 1999.

[6] Giles, J. "Internet Encyclopedias go Head to Head," Nature, 438 (531) www.nature.com/news2005/051212/full/438900a.html (Retrieved Dec. 13, 2009)

[7] Information Revolution Vs. Industrial Revolution - Locating Revolutions, Technology, Economics, Social Relations, Work, And Demographics, Continuing Debates, http://ecommerce.hostip.info/pages/578 /Information-Revolution-Vs-Industrial-Revolution.html, retrieved December 13, 2009.

[8] Landes, D. S. The Unbound Prometheus : Technical Change and Industrial Development in Western Europe from 1750 to the Present. 2nd ed.. New York : Cambridge University Press, 2003.

[9] Li, C., Bernoff, J., *Groundswell: Winning in a World Transformed by Social Technologies*. Boston, Ma.: Harvard Business Press, Forrester Research Inc., 2008.

[10] Mantoux, P. The Industrial Revolution in the Eighteenth Century, First English translation 1928, revised and reset edition 1961

[11] Micek, D.C., Micek, J.P., Whitlock, W., *Twitter Revolution: How Social Media and Mobile Marketing is Changing the Way we do Business & Marketing*. Las Vegas, NV: Xeno Press, 2008.

[12] MIT Open Course Ware, http://ocw.mit.edu/OcwWeb/web/home/home/index.htm, retrieved December 13, 2009.

[13] Palloff, R.M., Pratt, K., *Building Online Learning Communities: Effective Strategies for the Virtual Classroom*. San Francisco, CA: John Wiley & Sons Inc., 2007.

[14] Riel, M., Fulton, K., The Role of Technology in Supporting Learning Communities, *The Phi Delta Kappan*, 82 (7), 518-523, March 2001.

[15] Robertson, D.S., The Information Revolution, *Communications Research*, 17 (2), 235-254, April 1990.

[16] Sasson, D., New Teachers Use Twitter to Get Help and Advice, *Suite101* http://newteachersupport.suite101.com/article.cfm/new_teachers_use_twitter_to_get_help_and_advice, retrieved December 13, 2009.

[17] Tapscott, D., Williams, A.D., *Wikinomics*, expanded ed., Penguin Group: New York, NY, 2008.

[18] Twitter, http://twitter.com, retrieved December 13, 2009.

[19] "Twitter" *Wikipedia*

[20] Varian, H. R., Farel, J., Shapiro, C., *The Economics of Information Technology*. Cambridge, UK: Cambridge University Press, 2004.

[21] Vise, D. A., Malseed, M., *The Google Story*, New York, NY: Random House, 2005.

THE SIGCSE SUBMISSION AND REVIEW SOFTWARE:

10(HEXADECIMAL) LESSONS*

Henry M. Walker
Grinnell College
Grinnell, Iowa USA
641-269-4208
walker@cs.grinnell.edu

John F. Dooley
Knox College
Galesburg, Illinois USA
309-341-7748
jdooley@knox.edu

ABSTRACT

In 1999, SIGCSE 2000 Conference Co-Chairs, Boots Cassel and Nell Dale, asked Program Chair, Henry Walker, to develop the first iteration of a Web-based paper submissions and reviewing system. Due to the efforts of Ernie Ferguson, CCSC-CP was the first non-SIGCSE conference to use this system, and the system was used by CCSC-CP for 2006, 2007, 2008, and 2009. The current system for CCSC-CP 2010 utilizes the 11th version of this software. Each version has implemented refinements, new capabilities, and adjustments. Reflections on the evolution of this system yield numerous lessons for software and Web-based systems. This paper discusses 10 (hexadecimal) observations that can help in the design of future software applications.

1. INTRODUCTION

For SIGCSE 1999 and prior symposia, authors submitted several copies of their papers through the mail, and the Program Chair mailed copies to reviewers. By SIGCSE 1999, Program Chair, Robert Noonan, allowed reviewers to email their reviews, both papers and reviewers were entered into a database, and a series of scripts helped in the computation of ratings and tabulation of reviewer comments.

For SIGCSE 2000, Symposium Co-Chairs, Boots Cassel and Nell Dale, envisioned a Web-based system to give authors a choice of either hard copy or electronic submission

---

of their papers. Further, they hoped the submission of all reviews could be done over the Web. In response, Program Chair, Henry Walker, led a team that initiated an online system for the submission and review of papers. Since 2004, John Dooley has joined Henry Walker in developing and maintaining the software.

Since SIGCSE 2000, the software development has involved three major revisions and numerous refinements, and the system now exceeds 40,000 lines of code. Refactoring has become a way of life for this project, and much attention is paid to a clean design and structure when considering the refinement and expansion of new capabilities. Over the years, the system has experienced reasonable success, and in recent years it has been used by eight conferences (including SIGCSE symposia, ITiCSE conferences, SIGITE conferences, several CCSC conferences, and ACM-Southeast).

This paper reflects upon what the developers have learned from this process. Although some elements of the system are specific to individual conferences, the process also illustrates several general principles. In particular, this paper describes 10 (hexadecimal) lessons that may apply to a wide range of projects and system designs.

2. 10 (HEXADECIMAL) LESSONS

For convenience in exposition, the lessons are divided into four categories: diversity, streamlining routine requests for system-administration, maintainability and flexibility, and details and mechanics.

2.1 Diversity

From its beginning, the submission and review system was intended to serve the international computing-education community. Thus, the software needed to display material in a reasonable way, with good functionality, on a variety of computer systems (e.g., operating systems and browsers) internationally. Development of the system then facilitated broad involvement by both authors and reviewers around the world.

0. Expect browsers to render Web pages differently

Various browsers render html differently. Further, many browsers provide enhanced capabilities for the display of data. However, special functions that work beautifully on one browser may create awkward displays on other browsers; sometimes pages may appear blank on alternative browsers. To address this situation, we have made an on-going effort to follow the standards of the World Wide Web Consortium (W3C). When pages conform to W3C standards, browsers may differ on specifics of display but pages generally appear in some reasonable way.

Pragmatically, in the early years, getting software to work at all in the time required was a major challenge; for better or worse, we did not have time to consistently check W3C standards. In recent years, we have used the W3C validator at http://validator.w3.org to check various system pages.

Even with W3C standards, however, we periodically encounter bugs in specific versions of browsers. For example, the online program utilizes a range of formatting

techniques and is regularly verified for conformance to standards. However, a recent release of a common browser did not render one part of the standard properly, and the online program appeared blank on those browsers. In such cases, it has been necessary to rethink the submission and review scripts to determine a workaround for the problem - the pages must work reasonably, even if a common browser has errors.

1. Utilize a submission format that will look the same in all browsers

Beginning with SIGCSE 2001, the submission and review system has accepted only pdf format. For SIGCSE 2000, we accepted both html and pdf, but reviewers found that papers in html appeared in a wide variety of formats; authors could not be assured that reviewers would see a paper in a form related to what they thought they submitted. Some CCSC conferences have asked that the system utilize other formats, but numerous reports indicate versions of Word and other word processing packages appear differently in various computing environments. Thus, the current system does not accept these problematic formats.

2. Avoid JavaScript, or duplicate JavaScript at the server

The SIGCSE 2000 software relied upon JavaScript for error checking. The goal was minimize network traffic and to keep the load on the server low by doing checking at the browser. After a user completed a form, JavaScript running on the browser reviewed the data for completeness and obvious errors. Thus, at the server, we believed that we could assume submitted data was correct. However, during the reviewing process, we received numerous complains that review submissions yielded an error message.

Subsequent investigation suggests that many SIGCSE members, perhaps 30%-40%, turn off JavaScript on their browsers for security. Thus, if review form data contained errors or omissions, the browser would not detect problems, and the server did not work properly. We speculate that as many as a couple hundred reviews were lost for SIGCSE 2000.

In response, error checking now occurs at both the browser and the server. Users with JavaScript enabled will get quick error messages and warnings, as appropriate; and this remote processing can be done without network traffic and with no load on the server. However, since 2001, the same checks are repeated at the server, so errors are identified whether JavaScript is enabled or not.

3. Using Web-based system encourages international submissions and enables international reviewing

One great benefit of a Web-based system is that the submission and review of materials does not depend upon traditional mails. In particular, before SIGCSE 2000, reviewing was limited to North America - it simply took too long for papers to be mailed to Europe, South America, and Asia.

In addition, as part of an effort to increase membership participation within SIGCSE, the organization encouraged SIGCSE members to become reviewers for the

symposium (and later for the ITiCSE conferences). As a result, reviewing has become an extremely popular activity. For example, of the 1079 reviewers in the SIGCSE database in September 2009, 825 (about 76%) were from North America, and 254 (about 24%) were from outside North America. Reviewers could sign up to review for the symposium, the ITiCSE conference, both, or neither. Of the 845 available for the symposium, 680 (about 80%) were from North American and 165 (about 20%) were from outside. Also, of 686 available reviewers for ITiCSE conferences, 490 (about 71%) were from North American and 196 (about 29%) from outside. Also, since circumstances can cause a reviewer to be unavailable temporarily, 177 reviewers (127 - about 71% from North America) were inactive in September 2009.

This number of reviewers has allowed both recent SIGCSE symposia and ITiCSE conferences to send each submission to 6 reviewers to obtain a range of perspectives. Anecdotal evidence suggests the involvement of SIGCSE members with reviewing also has helped increase both conference attendance and SIGCSE membership.

Specific attendance figures are not available to compare CCSC reviewers and attendees, but on-line reviewing and on-line conference programs may encourage attendance at CCSC conferences as well.

4. A Web-based system enables both submitters and reviewers to wait until the last minute before sending in materials

Authors submitting papers before SIGCSE 2000 relied upon the postal system to deliver their materials; typically materials had to be postmarked by a deadline. However, variability of mail delivery encouraged early submissions, so authors were confident their submissions would arrive in time.

With Web-based systems, authors can take advantage of available time to review their work (or to write up articles at the last minute). In contrast to the early years, most submissions now come in very close to the deadline. For example, SIGCSE 2009 had Friday, August 29, 2008, as its submission deadline. Of the 305 papers submitted, 9 (3%) were submitted before August; 41 (13%) (including the earlier 9) were submitted over 1 week before the deadline; 264 (86%)were submitted during the last week; 51 (17%) were submitted the day before the deadline; 164 (54%) were submitted on the last day. Patterns for CCSC and other conferences seem similar.

2.2 Streamlining Routine Requests for System Administration

Day-to-day tasks of system administration are often straight forward. However, processing of numerous requests can take considerable time. To control the administrative work load, tools should be developed to allow users or conference leaders to handle many routine tasks.

5. Use a Permissions table to handle deadlines

The submission and review system includes a table within the database that specifies permissions. The table is divided into numerous fields corresponding to various

activities, and the table has just one record. If the record's field indicates "Yes" for a task, users are allowed to perform that activity. If the field indicates "No", users are not. For example, one field specifies whether submissions are allowed for papers, and the submission scripts reference this field. Before the submission deadline, the field is "Yes" and papers can be submitted or updated. When the deadline has passed, the conference leadership changes the field to "No", and submissions are cut off. With this approach system administrators need not be involved regarding decisions of deadlines, and policies can be set by the conference leadership.

To further enable leadership to take control, a Web-based administrative script allows conference and program chairs to change database fields. Thus, with a simple browser interface, any leader with Internet access can edit the Permissions table to change a field to "No" after a deadline passes.

6. Expect users to maintain their own data, and send confirmation notes for database updates

For SIGCSE 2000, authors either submitted their papers entirely online or submitted author/title data online, backed up by hard copy. However, the system had not advanced to allow them to change their data. Thus, after acceptances were mailed, authors were instructed to contact the administrator to revise fields (e.g., co-author names, paper titles, etc.) In all, 78 papers were accepted for SIGCSE 2000, and the Program Chair received in excess of 80 requests for changes; some papers required no changes in author/title data, but multiple requests were received for other papers.

Thus, for SIGCSE 2001 and for subsequent CCSC and other conferences, the submission and review system allowed both authors and reviewers to update their own data. Further, to verify changes, the update scripts send email to individuals when data are changed, so adjustments can be reviewed and refined further if necessary. From time to time, leaders still may need to make special changes, but the vast majority of the work of updates now can be done by the users themselves.

7. Assume submitters and reviewers will forget their ID numbers and passwords

In most years, the most common question asked to system administrators involves forgotten numbers and passwords. Authors misplace email with their paper numbers, reviewers forget their reviewer numbers, and many users forget their passwords.

To resolve such difficulties, at least two solutions are possible. In the early years, an administrative script was developed to look up author and reviewer data, based on either last name or possible ID. With this script, many email requests could be answered, although in practice information would only be sent if the email in a user's record matched the email request.

More recently, reviewers are able to get their password information by identifying themselves in a special form. Email information is sent to the email address on record in the database. Of course, if email addresses are misspelled, the administrator still may receive email requests for ID/password information from users.

With either approach, it is vital for administrators to be able to respond quickly and easily to requests for forgotten IDs and passwords.

2.3 Maintainability and Flexibility

A software system is useful only if it addresses actual needs. However, hardware and software sometimes go down, the user community changes over time, and expectations evolve.

8. Use symbolic links for early, regular, late, system-down pages

For a typical conference, work on the submission and review Web site involves several stages. First, scripts are ported to a new directory, the database is cleaned from any past conference(s), and style sheets and details are adjusted to reflect the new event's look and feel. Next, the site is tested, to be sure it works as envisioned by conference leaders. Then, the site is open for active use. Finally, deadlines pass, and submissions and reviews are no longer accepted.

With this process, it is helpful to have a general "site under development" page at the start, live scripts for database processing in the middle, and a "submissions are closed" site at the end.

In addition, users need to be notified in the case that a database server is down.

Although each of these capabilities involves different pages, submissions and reviewer registration should be accessible through a single Web URL. For example, the general conference pages should list submission.shtml, and this reference should not change over the duration of the conference.

Symbolic links are particularly useful in this context. Conference pages specify links for submissions and reviewer registration, submission.shtml and reviewerRegistration.shtml. In practice, both of these files are symbolic links. During site development, submission.shtml refers to an "under construction" page, submissionEarly.shtml. This link is changed to active submission scripts when papers are accepted. The link is changed a third time to submissionLate.shtml when all submissions are closed. In addition, submissionDown.shtml notifies users if the database software or platform is malfunctioning.

9. Expect numerous suggestions, many of which will be contradictory

As the system has evolved, conference leaders have changed, and users have expanded their expectations. Thus, over the years, we have received a steady stream of suggestions, and this feedback is greatly appreciated! As a practical matter, however, users have different perspectives, and ideas may not be consistent. Further, the system depends upon some philosophical and policy decisions. With such diverse viewpoints, it would be easy for tinkering with the system to yield chaotic code over time. Adjustments in one place could easily undermine the database or script processing elsewhere.

With so many suggestions, therefore, it has been essential to maintain a clear, conceptual view for both the design and implementation of the system. If one is not careful, a simple technical fix has the potential to solve one problem, but create many more. To allow long-term reliability and maintainability, it has been essential to assess each potential refinement within the overall structure.

Altogether, we have been delighted to accommodate ideas and refinements, when these viewpoints seem consistent with the overall framework of the system. However, brainstorming is needed to find alternatives, when philosophies collide.

A. Don't underestimate the value of documentation!

Conference leaders must tackle dozens of tasks in their planning, and much of this work involves handling submissions, reviews, acceptances/rejections, and the final program. Although we have tried to support these tasks with the submission and review system, tasks for the leadership are often detailed. Further, leaders rotate in their responsibilities, and new leaders are involved on a regular basis.

To help leaders, particularly new leaders, work through their numerous activities, documentation is essential. Leaders must know what tasks they need to do, and they need guidance in how that work might be done. Current documentation, therefore, provides considerable detail for numerous tasks - now extending to about 50 pages. Of course, keeping the documentation in synchronization with system refinements is a constant challenge. However, most day-to-day questions received from conference leaders relate to incomplete or outdated documentation. When documentation is current, clear, and complete, conference activities seem to go rather smoothly (at least from the standpoint of electronic submissions and reviews). Interested readers can find documentation for CCSC CP-2010 at http://www.cs.grinnell.edu/~ccsc/ccsc-cp2010/documentation.shtml.

2.4 Details and Mechanics

The submission and review system has evolved remarkably over its 11+ years of existence. Several low-level details and capabilities have made a significant difference in the systems evolution and use.

B. Organize database tables conceptually rather than by common data

For historical reasons, the SIGCSE 2000 system contained one table for people; the same table was used for both authors and reviewers. Likely, this common table saved much manual data entry.

However, early experience with the Web-based system indicated that different information was needed for submitters and for reviewers. A careful review of fields indicated that some data were needed for both groups, some only for submitters, some only for reviewers, and some stored data were never used.

In addition, over time, people might use multiple email addresses, and people might change institutional affiliations. Also, in moving from one conference to the next, reviewers should continue in the database, but not authors (until they write another

paper). Altogether, it was extremely difficult, perhaps not even possible, to match a reviewer with an author; and data from one year might or might not relate to data the next year.

With these observations, the database was redesigned for SIGCSE 2001. Authors and reviewers were conceptually different, even though they had some common fields, and their data were used differently. In retrospect, this adjustment simplified many later refinements of the system. When database tables match a clean, conceptual model, the design of scripts can proceed reasonably smoothly.

C. Use include files/style sheets/table-driven options

In writing the SIGCSE 2000 scripts, the design team was thrilled to develop working scripts within a tight time frame. Also, with limited experience, it was difficult to anticipate how a system might evolve in the future. As a result, elements of the look-and-feel of the conference were hard coded into the scripts, as were conference dates and leaders' names.

Of course, for SIGCSE 2001, these details had changed. Thus, at an early stage, header files were used to specify conference logos. Further, a separate file was developed with variables to handle numerous conference details (e.g., conference name, year, location, dates). These pieces allowed administrators to change just a couple of files when moving from one conference to the next.

Within a few years, conference Web sites had become more sophisticated, with style sheets specifying a consistent look and feel; and style sheets too became part of the submission and review system.

Within another few years, leaders wanted to add panels and special sessions to online submissions, in addition to papers. Leaders for CCSC and other conferences identified still more submission categories. For new categories, scripts could use much of the capabilities developed for papers, but wording required adjustment. For awhile, duplicating paper scripts with edits provided a solution. However, with the addition of workshops, birds-of-a-feather, and other categories, the approach of separate software for new categories was clearly unrealistic.

A nicer solution has involved the creation of a new table to handle submission types and form options. Fields in this table provide extensive data on numerous choices and options for each category of submissions. For example, one field involves submission deadlines, another the name of the table used to store submitter information, and another the type of review form to be used. Once established, scripts can tailor the display of forms, gathering of data, and processing of information according to the parameters specified for each category of submission. As a side effect, the online program can adjust the presentation of sessions, according to additional category parameters.

D. Have common script for submitter or reviewer or administrative login

Much processing requires authentication. For example, only an author, designated reviewer, or conference leader should be able to access a paper. General conference

leaders should have access to any part of a system, but panels' or workshops' chairs can only access submissions in their purview.

To simplify authentication for much processing, a common script handles this work. The authentication script checks passwords and sets underlying variables as needed. Other scripts simply include this function for protection. This approach requires moderate care in writing the basic script for authentication. However, including this script elsewhere provides a reliable mechanism for enforcing security through much of the system.

As an additional feature, once an administrator, conference leader, or user logs in successfully, the authentication script can set environment variables to remember this success - at least for a short time. Thus, processing can go from one task to the next without the need for constant logins - until the time limit expires.

E. Sending email requires flexibility and options for testing

Conference leaders must send periodic communications to both submitters and reviewers. Submitters need to know about acceptance or rejection; and authors of accepted papers must know about forthcoming deadlines and procedures. Reviewers need to know their reviewing assignments, and they need reminders and status reports as reviewing deadlines approach.

Each email will likely draw information from the database within the context of other text. The submission and review system accomplishes this task by allowing leaders to upload a template for a letter. Within the template, scripts insert designated variables from database records. For example, when the variable $rev-assign appears in a template for reviewers, a script includes a list of submission assignments for each reviewer before sending email.

Once a template is prepared, leaders need to review sample letters before they are sent. Thus, the email scripts operate in two modes. Testing mode (the default) composes full letters, but sends them to the leader (for checking) rather than the submitter or reviewer. "For real" mode uses the actual email addresses of users.

In addition in sending email, some letters go to specific categories of recipients. For example, authors of accepted papers should receive different letters than authors of rejected papers. For both submitters and reviewers, leaders can choose distribution among several groups; and over the years, the range of distribution groups has expanded greatly.

In the actual sending of email, the system relies on a departmental email server. Although this normally works well, the server occasionally becomes overwhelmed or otherwise shuts down. To resolve any difficulties in this sending of email, the scripts allow leaders to send email to authors or reviewers with IDs in a specific range, and a list of each recipient is send to the Web interface as an email is sent. By default, email goes to all people with any ID. However, if a leader wants to send a test message to just a few people, the leader can specify just a few people (e.g., ID 1 to 10). Also, if email starts, but the server stops for any reason, the Web list shows how far the process went, and the next batch of email can be restarted where the last left off.

F. Refactoring not only aids maintenance, but also helps standardization of elements of forms (e.g., subject display, country display)

Common software practice suggests that developers should constantly review their code to find common activities. These tasks then can be abstracted into a library of common functions that can be used throughout the application. Such refactoring can aid reliability and maintenance.

For Web-based applications, refactoring also can provide substantial assistance in standardizing forms and user interfaces. For example, forms for both submissions and reviewer registration ask for an individual's country. By placing this part of a form within a function, all forms are guaranteed to utilize the same codes and have the same appearance.

Similar results apply to the display of subjects identified by submitters or by reviewers. In this case, however, the abstracted design requires considerable care. In particular, subject areas, such as "not appropriate for this conference", are not relevant for authors, but reviewers might find this designation appropriate. Similarly, some subject areas might apply to authors, but not reviewers. Finally, some subject areas might apply to some submission categories (e.g., videos), but not others. All these combinations can be organized into a nicely refactored function, but considerable care is needed in finding a clean and elegant design.

3. CONCLUSIONS

The current submission and review system has helped facilitate the transition from a paper-based system to a Web-based interface for several categories of conference submissions. Reflection on the evolution of this system may inform the design of other systems.

The system for SIGCSE 1999 began when Web-based applications were just emerging, and the authors hope that subsequent systems have been helpful to the community. In recent years, more general submission and review systems have begun to emerge, and it is possible that these may supersede the current SIGCSE/CCSC/SIGITE/ACMSE system. Regardless of the nature of future software, the last decade of experience provides lessons for software development that may be useful in a range of applications.

4. ACKNOWLEDGMENTS

The authors express their sincere thanks to the many people who have contributed to the development of this system over the past 12 years. SIGCSE 2000 Co-Chairs Boots Cassel and Nell Dale provided the initial motivation for the project, and students Dorene Mboya and Weichao Ma worked with Henry Walker on the first version, with extensive guidance and technical support of Wayne Twitchell and Theresa Walker and funding from Grinnell College.

In the subsequent 11 years, Grinnell College has continued to support the servers and programming environment. Through their collaboration, the authors also gratefully

acknowledge the many suggestions from Symposium chairs, the SIGCSE community, and leaders from CCSC and other conferences.

A CAPSTONE EXERCISE FOR A CYBERSECURITY COURSE*

J. R. Aman, Ph.D.
Saint Xavier University
Chicago, IL 60655
aman@sxu.edu

James E. Conway, CISSP
State of Ohio
Columbus, OH
james.e.conway@gmail.com

Christopher Harr, MACS
Saint Xavier University
Chicago, IL 60655
harr@sxu.edu

ABSTRACT

The design and execution of the capstone project in a computer security course is described in this paper. The course followed a 'black hat/white hat' philosophy, teaching both offensive and defensive techniques and tools. The project required students working in small teams to apply awareness of common system weaknesses and knowledge about available tools for attack and defense to locate several "flags" within a network in a limited time frame. The target network was accessed through a virtual private network (VPN), which provided a measure of security to the university and secured the connection between the students and the target.

BACKGROUND

At the 2005 SIGCSE Symposium, the session on computer security sparked a lively exchange between two presenters and actively engaged the large audience. John Aycock's and Ken Barker's (University of Calgary) paper offered the rationale for a course on computer viruses and malware. The course had caused some controversy because students were taught both how to create viruses and how to defend against them, a so-called "red team" approach. Patricia Logan (Marshall University) followed with a paper [2] (co-authored by Allen Clarkson) which, while also supporting the teaching of

attack-based techniques, emphasized the need for other issues to also be covered. First among these was ethics. Others included appropriateness of course content, university security concerns, and security lab design. What made the session stand out was the eloquent and well-reasoned opposition Dr. Logan took to the Calgary approach and the interaction which developed between the two presenters. The discussion was enlightening, energetic, and collegial. Audience reaction visibly swayed between the two with no clear consensus for either position.

After considering what had been heard at that presentation, an attack-based -- or 'black hat/white hat' -- approach became the format of a graduate cybersecurity course at Saint Xavier University. The Department of Computer Science places an obvious emphasis on ethics in every course in its curricula at both graduate and undergraduate levels. That met one of the concerns Logan had expressed. Questions of lab design and the security of the campus networks were answered through close consultation with the university's Systems Group. Thus, the course has blended the best reasoning of both presenters in the SIGCSE session.

The course has now been taught three times in this attack-based format. Students and faculty refer to the course jokingly as "the Dark Arts course." Details of the course curriculum have been previously described [3]. The culminating project, however, was changed significantly for the third course offering and is described in this paper. That project would be equally applicable to an undergraduate course.

The time invested in creating and implementing a computer security course and the projects in it is not trivial, as anyone who has done so will undoubtedly testify. Hardware, software, logistical, and institutional security considerations abound. Tikekar and Bacon [4] described these challenges and offered their perspective on handling them in an excellent 2003 paper. Yu [5] adds additional practical details, such as rules for avoiding potential illegal activities. The *ACM Journal of Educational Resources in Computing* devoted an entire special issue in 2006 to resources for such courses [6]. The lead editorial laid out the case and highlighted many of the issues Tikekar/Bacon and Yu also identified.

In the following sections, the exercise is described, then details of the hardware and software setup are outlined. Responses of students and staff are noted and plans for further development are identified.

THE CAPSTONE EXERCISE

The capstone exercise -- called "ByteMe! 2008" -- was designed primarily by a CISSP-certified security specialist who served as consultant to the instructor and the students. The final six weeks of the course were devoted to this effort.

Week(s)	Content
1	Meet with consultant
2-5	Locate the 5 flags
6	Submit written & oral reports

The initial meeting with the consultant involved a comprehensive presentation about the cybersecurity field as a profession, the challenges faced by security specialists, the need for corporate planning for prevention and response, and the importance of ethical practices. Then the project itself was introduced, teams were identified, the "rules of the game" were discussed, and specific technical details were distributed. At the end of the presentation, the clue to the first flag was released.

The rules were simple and straightforward:

1. Never use any tool outside the VPN connection!
2. Maintain a log of all activities to be submitted at the end of the exercise. Include all pertinent data (date, time, person conducting test, tool/activity, arguments applied, results obtained, intended goal, etc.)
3. Log details of all vulnerabilities found along with suggestions for hardening
4. Goal: Locate and record the textual contents of five (5) "flag" files located at various depths in the Acme Widget network
5. Last chance to complete the exercise: <date inserted here>
6. Report and submission of logs: <date inserted here>

Although it was possible to work outside the department's lab, most students confined their work to that facility to avoid possible problems with external ISPs. There were spirited attempts to socially engineer information among the groups, and sometimes they were successful. Overall, however, the groups maintained a level of privacy which surprised the instructor. Not until the final week was there any apparent move toward a concern for the group as a whole. In the closing hours, some people from successful groups did mentor struggling groups.

DETAILS: HARDWARE/CONNECTIVITY

The department's lab workstations were dual-boot Windows XP Professional and SuSE Linux. A virtual private network (VPN) connection was established with the consultant's hardware using Checkpoint's VPN-1 SecureClient. SecureClient is Windows-based so the Linux tools were not available. This was taken into consideration in the placement of the flags. Using a VPN client allowed the students to securely connect to Acme Widgets without interference to the university's network. Checkpoint's client was an easy, small install that required administrator privileges on the workstation. Students were given a copy of the installer program for use on their personal computers. This allowed them to work from outside the lab, although most did not, as previously mentioned.

Checkpoint's client is supposed to route all traffic through the common TCP port 80, however there were problems with the campus's Cisco firewall. The university's Systems group was unable to pinpoint which port the SecureClient was trying to use but they were able to add a rule to the Cisco firewall to allow all traffic to and from Acme Widgets. Most students were not running an equivalent Cisco firewall outside the lab and did not have connection problems. Another problem which surfaced was that Checkpoint's VPN server was software based; students would regularly crash the Microsoft operating system running Checkpoint's VPN, dropping all VPN connections until the server was reset. The consultant was busy on class nights 300+ miles away!

DETAILS: COMPANY/FLAG SETUP

Details of the Acme Widgets domain are much too long to include completely in this paper. The consultant created the system in its entirety. Some details will be given here. For a full listing, contact the primary author. A complete list of Acme's users, their passwords and privileges, their corporate responsibilities, and the domain groups are available.

Acme Widgets ran five virtual LANs, one each for management, DMZ servers, production workstations, production servers, and a test lab. There was one firewall, a Cisco 5000 switch, and a Cisco 1900 switch. The DMZ held a Windows 2000 server (Foghorn) running IIS with Apache and a Suse Linux server running Apache. Flag #3 (Oracle.hlp) was on Foghorn.

The production workstations were a virtual host server (WorkMaster), a Windows XP machine with no service packs in place, and a Windows 2000 workstation. The latter held Flag #1 (WhiteRabbit.flg).

The production server farm held two Windows 2003 servers (Wiley and Nosce), a Windows 2000 server (RoadRunner). Three of the flags were in this farm: #4 (Hello.neo) on Wiley, #5 (KnowThyself.txt) on Nosce and #2 (Trinity.db) on RoadRunner. Details of the flags are appended to this paper.

RESPONSE

Students echoed the concerns about SecureClient noted already. Their comments about the course, including ByteMe! 2008, were along the lines of these representative examples:

- "It demanded that we continue to try every tool, even if it took more than 10 times."
- "Forces student to rely on persistence to be effective"
- "Allows us to use different software tools which are useful and can be handy"

The instructor, consultant, and lab director have reviewed the course and the project at length in preparation for the next scheduled offering in spring, 2010. We are pleased with the enthusiasm generated by the exercise but are also keenly aware that a defensive component is needed. Students wanted more hands-on work with the tools prior to the capstone exercise so very targeted exercises are being written to address this from both the black hat and white hat perspectives.

SUMMARY

Although a live penetration audit had been used in prior years, this directed penetration was received with much more enthusiasm. To be sure, trying to compromise a firewall and workstations within a LAN are interesting and can be exciting. However, students preferred the more game-like environment of the flag hunt. Anecdotally, this had much to do with knowing the flags existed. In the previous exercise, there was no assurance anything would be found at all. Also, the clues and the nature of the pathways to the 'flag' files constituted a puzzle which students seemed to enjoy.

The flags were all placed to emphasize configuration errors by a network or system administrator. There was no attempt to be particularly devious. Nonetheless, students initially over-thought the possibilities instead of looking for obvious faults. Once they realized what was happening, they moved steadily through the flag sequence. That is why several hints were needed to guide teams to the first flag (WhiteRabbit).

Overall, the results were very encouraging. Clearly, students enjoyed the challenges. Because they had known a difficult task lay before them in the closing weeks, they had been diligent in learning to use the tools.

From the instructor's perspective, students had to combine their understanding of networks, servers, and common system administration errors with working knowledge of the penetration tools and exploits available to them. Finding the flags required a great deal of knowledge synthesis as well as technical application. This project in this course served as a focal point for knowledge acquired through other courses in the curriculum. As the course and the project are revised and, hopefully, strengthened over the next year, even greater emphasis will be placed on these related areas and on critical thinking processes.

What was missing in this exercise was any experience hardening systems. This could be accomplished with a more traditional "red team/blue team" or "capture the flag" exercise. However, this was a very good exercise. We are working now to add a "white hat" component to ByteMe! 2010.

The VPN server was subject to frequent crashes. This was the major problem during the in-class sessions of the exercise weeks. Occasionally, several students working on their own had managed to crash the VPN server. This was more problematic, because the consultant could not reboot until reaching home again. In the future, the VPN server will either be separated or a hardware-based VPN server will be used.

APPENDIX: THE FLAGS AND HINTS

#1: WhiteRabbit.flg

Trinity hacked the IRS database through RoadRunner. The answer is out there, and it's looking for you, and it will find you if you want it to.

Hints given as guidance to the first flag:

Hint 1: Flag is found in the 172.18.50.0 network (the production segment)
Hint 2: Don't be shy to be a guest in this network
Hint 3: Some passwords are non-existent
Also remember to use the ENUM program (or something else of your choosing) to get listings of directory contents!

#2: Trinity.db

You have to let it all go. Fear, doubt, and disbelief. Free your mind. When the fog clears, the oracle will be revealed. Remember, there is no spoon.

#3: Oracle.hlp

> Oh, what's really going to bake your noodle later on is, would you still have found it if I hadn't said anything? You must be cunning and wily to finish your journey.

#4: Hello.Neo

> Do you know what "Nosce te ipsum" means? It means know thyself.
> You have to trust me. You've been living in a dream world.
> Go to room 303. The answer is available if you look beyond the physical.

#5: KnowThyself.txt

> You've done it. You've found it. Welcome to the real world.
>
> Additional hint for Flag #5: There are alternate homes for data in obscure streams of presence hidden from all but those who research carefully.

REFERENCES

[1] Aycock, J, and Barker, K. Viruses 101. *ACM SIGCSE Bulletin*, 37 (1), 2005.

[2] Logan, P., and Clarkson, A. Teaching Students to Hack: Curriculum Issues in Information Security. *ACM SIGCSE Bulletin*, 37 (1), 2005.

[3] Aman, J. Black Hat/White Hat: An Aggressive Approach to a Graduate Security Course. *Journal of Computing Sciences in Colleges*, 22 (2), 2006.

[4] Tikekar, R., and Bacon, T. The Challenges of Designing Lab Exercises for a Curriculum in Computer Security. *Journal of Computing Sciences in Colleges*, 18 (5), 2003.

[5] Yu, Y. Designing Hands-On Lab Exercises in the Network Security Course. *Journal of Computing Sciences in Colleges*, 22 (5), 2007.

[6] Frincke, D., Oudekirk, S, and Popovsky, B. Editorial: Special Issue on Resources for the Computer Security and Information Assurance Curriculum: Issue 1. *ACM Journal of Educational Resources in Computing*, 6 (3), 2006.

AN INTRODUCTION TO VERSION CONTROL WITH SUBVERSION*

TUTORIAL PRESENTATION

John Cigas
Department of Computer Science, Information Systems, and Mathematics
Park University
Parkville, MO 64152
cigas@acm.org

This workshop introduces participants to Subversion, a version control system. Participants will work with an established repository (containing a web site) so they can focus on key, daily-use concepts, such and committing, updating, resolving conflicts, branching and merging. Using a web site model is familiar and allows viewing of all changes during the workshop exercises. Additionally, we will demonstrate installation of a Subversion server and creating a repository as time allows. This workshop is for people completely new to using a version control system or for those new to using one in a group environment where conflicts will occur. A laptop is required. Linux, Mac and Windows will all work. Participants should download a Subversion client before the workshop so they'll be ready to hit the ground running.

One of the key headaches with learning a version control system is setting up the repository. This workshop alleviates that problem by starting with an existing repository. Once participants have a better understanding of using the entire system, then they can move on to setting up their own repository. The other piece missing when learning a system like Subversion is dealing with other people's changes. Since there will be multiple people in the workshop, we will have people working independently on the same files and the exercises will force participants to see how these issues are handled.

IMPLEMENTING AN INTERDISCIPLINARY CAPSTONE EXPERIENCE FOR INTERACTIVE DIGITAL MEDIA MAJORS*

Carol Spradling, Jody Strauch
Computer Science/Information Systems, Mass Communication
Northwest Missouri State University,
Maryville, MO 64468
1-660-562-1588, 1-660-562-1823
c_sprad@nwmissouri.edu, jody@nwmissouri.edu

ABSTRACT

In this paper, we describe an interdisciplinary Interactive Digital Media Seminar capstone course at Northwest Missouri State University, which is co-taught by a team of faculty from three departments; Computer Science/Information Systems, Mass Communication, and Art. The Interactive Digital Media majors of Computer Science Programming, New Media and Visual Imaging, take this seminar course during the fall semester. This paper describes the course structure and the process utilized to develop and teach this interdisciplinary seminar course and shares the formal assessment process and observations about working with an interdisciplinary team and teaching students from the three majors.

1. INTRODUCTION

There is a long tradition to include some type of capstone course for a major. Most capstone courses require some type of work that is supervised by an instructor and provides a culminating and integrative education experience for students [3]. While capstone courses vary from major to major, the capstone experience may include team-based projects, research papers, field projects, portfolio projects, oral presentations, group work, multimedia presentations and internships. Some courses work with clients, while other courses may incorporate individual student work.

Most capstone courses require that students have completed a set of courses as a prerequisite for the capstone course. Students are expected to apply previously gained knowledge from these prerequisite courses, reflect upon this knowledge and prepare some type of artifact that demonstrates the students are prepared for future challenges in their field of study.

The majority of capstone courses, however, are mono-disciplinary, that is, they are capstones of degrees housed within a singular discipline. New challenges and opportunities can emerge in an interdisciplinary capstone course. In a degree program of courses taken from a variety of disciplines (a multidisciplinary degree), an interdisciplinary capstone offers students the opportunity to not only integrate what they've learned from previous courses but also incorporate the perspectives of other disciplines involved in the same degree. Woods noted that "socialisation into a discipline subtly shapes ways of thinking and orientations to learning and that can ultimately lead to mutual incomprehension when specialists from different subject domains try to collaborate." [9, p. 854] She noted that an "interdisciplinary communication competency" requires both knowledge components, i.e. terms, technology, etc., as well as cultural components, such as "awareness of how perception of other disciplines than one's own can affect interaction." Woods believes the development of this type of interdisciplinary communication is as important as the integration and application of discipline-specific skill sets learned in previous courses.

One might argue that teaching an interdisciplinary capstone course will provide an opportunity for students to implement Boyer's scholarship of integration and application [2]. For a multidisciplinary degree program, only a true interdisciplinary capstone course can provide an opportunity for students to implement Boyer's ideals.

The interdisciplinary capstone course will provide both faculty and students the opportunity to make connections and gain insights across several disciplines, in this case, the disciplines of computer science, mass communication and art. Boyer and Scherpereel suggest that "cross-disciplinary teaching requires faculty who are knowledgeable or at least comfortably familiar with, the other course disciplines, to provide consistent guidance for student learning" [1, p. 224].

Interdisciplinary capstone courses place special demands upon instructors and the students. Our intent is to provide an interdisciplinary studio-based experience that allows students to apply their diverse knowledge and expertise to a set of class activities, and one large electronic portfolio project. Additionally, students must have an opportunity to develop their technology, design and communication skills in the context of their discipline. The instructors' role in this course is a challenge because student demands vary from student to student. The instructors are required to mentor students and use their technical expertise (often gaining new technical expertise) to assist students as they develop the electronic portfolio and research topics within their particular discipline. This process can be very time-intensive for both the faculty and student.

This paper will discuss the working issues surrounding our endeavor to develop and teach an interdisciplinary seminar capstone course for Interactive Digital Media majors from three disciplines. Section 2 provides background information, Section 3 presents information about our course structure, Section 4 discusses assessment, and Section 5 shares our lessons learned through course changes, impact of the course and challenges.

2. BACKGROUND INFORMATION

2.1 Major and Course History

Northwest Missouri State University is a state-assisted four-year, moderately selective mid-western university with approximately 6,100 full-time equivalent (FTE) undergraduate students and 1,000 graduate FTE students. The university offers 100 majors, 70 minors and 21 pre-professional programs.

The Bachelor of Science in Interactive Digital Media major was approved by the state Missouri Coordinating Board of Higher Education in 1999 and the first students entered the program in the spring of 2000. The major is shared by two colleges, Business and Professional Studies and Arts and Sciences, and requires 124 hours with 42 hours of general education courses. In particular, the Interactive Digital Media is a 61-hour multi-discipline major composed of core courses from the departments of Computer Science/Information Systems, Mass Communication, and Art. The core consists of 36 hours and the student must choose one of three concentrations with 25 hours in each: Computer Science Programming (Computer Science/Information Systems), New Media (Mass Communication), or Visual Imaging (Art) [8].

During the 2000 - 2003 years of the Interactive Digital Media major, students were required to take a departmental seminar course offered by each of the individual departments of Computer Science/Information Systems, Mass Communication or Art. However, since the fall of 2004, a one-hour team-taught interdisciplinary seminar course has been offered. Students enroll in the Art, Computer Science/Information Systems or Mass Communication section of the Interactive Digital Media Seminar course. All three sections of this course meet at the same time and location and meet in either a conventional classroom or a computer laboratory. Students enrolled in a particular course have an instructor assigned by the particular department of their concentration; however, students complete the same assignments and work collaboratively on some assignments.

2.2 Student Profiles

While we projected 45 students between the three concentrations for the first year and projected 120 students after five years, the popularity of this major exceeded our expectations. Enrollments reached 130 students in the 2008-2009 academic year. For the fall 2009, our majors between the three departments are 110 students. Twenty-four students declared a Computer Science Programming concentration, 47 students declared a New Media concentration, and 39 students declared a Visual Imaging concentration. During these past six years, approximately ten students have declared double concentrations, which we did not anticipate.

3. COURSE STRUCTURE

3.1 Course Information and Prerequisites

The Interactive Digital Media Seminar is a one-hour course which meets weekly. Although students are enrolled in separate sections within each of the departments, instructors and students meet for one hour and fifteen minutes each week for the entire

semester. Occasionally, instructors meet with students from their particular concentration (Computer Science, New Media or Visual Imaging) for short meetings to discuss topics of interest to the particular concentration.

The Interactive Digital Media Seminar is usually taken during the senior year. Prior to taking this course, students are required to take the core courses and will have taken most of the courses in their concentration. Although we do not have formal prerequisites, most students have completed the core courses prior to the seminar course and are taking courses from their concentration during their senior year.

3.2 Course Goals

Students enrolled in this capstone seminar course will seek jobs in interactive digital media areas which may include: media production, design/layout, web interface systems, content development, Internet technical programmer, webmaster, or project manager. Therefore, the goals of our seminar course are (1) to provide a means for students to integrate their prior knowledge into the development of a large professional electronic portfolio, (2) to allow students to develop confidence in their ability to work on a multimedia project, (3) to work in a multidisciplinary team as they will in a professional setting and learn to build a learning community, (4) to become familiar with and conduct current literature searches on interdisciplinary industry topics, (5) to develop professional skills which will be used to search for a job in their industry, and (6) to develop skills that will allow students to enter the workforce for their discipline. With these goals, we want to produce a rich and meaningful experience for students that will require a combination of technical expertise, communication skills, visual aesthetic skills, and creative abilities.

3.3 Pedagogical Methods

Studio-based approaches are well-documented in the arts and architecture fields, but are relatively new to the field of computing [4]. The studio-based approach includes some type of design problem, collaborative learning usually with critiques by a master teacher and the student, and some form of an exit interview. The model allows students to have a community of learners that interact and solve problems. Because the Art and Mass Communication faculty use this pedagogy in their design classes and students have previously been exposed to this learning model, it seemed natural to continue this pedagogy for the interdisciplinary capstone seminar class.

3.4 Course Information and Assignments

Student assignments during the semester consist of (1) units to allow the students to develop their "soft skills" by creating a business resume, a list of references and a cover letter, interviewing for jobs in the student's particular profession with a follow-up of a required mock interview with an industry professional and a requirement to attend a career fair, (2) some form of research on a current industry topic, (3) the creation of an electronic portfolio using either Flash, HTML, CSS, JavaScript or other technologies, and (4) a job shadow in which students must have their portfolio critiqued.

We require students to write reflection papers on the mock interview and career fair experiences. Collaborative team projects and individual project pedagogies have been utilized over several years for the current research topic assignment. The student job shadow assignment and portfolio review are two requirements with an industry professional. Students may complete these requirements simultaneously or with two separate professionals. Students are required to write a report on this experience.

The individual electronic portfolio includes samples of students' course work and client work in the three areas of design, programming and project management. The portfolio project has always involved interdisciplinary, collaborative faculty and student-based critiques at several stages through the portfolio project. Additionally, over the years, we have learned to break up the portfolio project into several stages and established that firm deadlines are required of students. Currently, we include seven stages in the electronic portfolio project: (1) a portfolio plan that describes how students will approach their portfolios, including the focus of their portfolios, (2) a list of all the project files that will be included in the portfolios, (3) the portfolio design including the "look and feel" for their portfolios and some storyboarding, (4) a portfolio prototype that includes two working sections, (5) interdisciplinary team critiques of the portfolios, (6) a critique by members of their particular discipline, and (7) a final oral presentation of their portfolios to the entire class.

3.5 Instructor Collaboration

Faculty collaboration has taken on many forms from determining the goals and content of the course to the development of assignments and assessment to the expertise of faculty on a particular topic. The faculty collaboration has increased our ability to develop a variety of materials for the course, our motivation to try risky topics, and encouraged creative approaches to technology, pedagogy and assignments. Because of the interdisciplinary faculty backgrounds, each instructor contributes a vision of what we want our students to experience in the capstone seminar course, distributes his/her faculty expertise and then operates as a mentor or facilitator to the students. The faculty collaborative role is certainly more demanding than that of a traditional course.

Our process has been to brainstorm and then settle on an approach for a technology, assignment or pedagogy. Over the past four years, we have experimented with team projects, industry speakers from the various disciplines, instructor and student critiques, researching industry topics, and electronic portfolios with built-in critiques from students, faculty and industry experts. Each semester, we refine the course assignments and content.

3.6 Student Collaboration

Most students have been exposed to educational systems that stress individualized learning which is provided by the instructor. Student assessment is then measured on the individual knowledge that students have gained. Student collaboration, on the other hand, stresses the collective discovery of knowledge and rewards and assesses the group rather than the individual, which is a difficult transition for many students. The arts have a long

tradition of studio-based collaborative critiques, and because the Interactive Digital Media students take several art classes, they come to this course with collaborative studio-based critique experiences of their design projects [4].

Student collaboration has many benefits. (1) It reduces the amount of design time, (2) provides support for students with both technology and aesthetics as they work in a team, and (3) allows them to partner new ideas within a team. While most assignments in our capstone seminar course are individual assignments, we use student collaboration to enhance the student's individual assignments.

4. ASSESSMENT

4.1 Student Teams and Assessment

Students are broken up into interdisciplinary teams composed four or five students for electronic portfolio critiques. Team selection is mixed with students from each of the three concentrations, Computer Science Programming, New Media and Visual Imaging. Instructors form interdisciplinary teams based upon students' technical, design, communication and social abilities to ensure that the teams are well balanced.

Students break into teams and assess student work at three critical phases in the electronic portfolio process, (1) the "look and feel" or storyboard phase, (2) the interdisciplinary team critique toward the end of the portfolio process, and (3) the concentration critique in which they are assessing work from students in their discipline toward the end of the portfolio process. For the interdisciplinary and concentration phase, the instructors provide the students with a checklist of criteria of verbal and numerical student-based criteria to evaluate each other's portfolios. The interdisciplinary studio-based critique phase is especially beneficial for the students as they collaborate and suggest different approaches to presenting their portfolio work. Students are asked to evaluate their team members based upon a checklist provided by the instructors. While this task is difficult for students, we have determined that it is beneficial for them to evaluate each other on each student's contributions to the team.

4.2 Instructor Assessment

Instructor assessment is intensive and takes place in several places in the course with individual assessment used for some assignments and interdisciplinary studio-based critiques for other assignments. First, individual feedback is provided by each instructor to students in their concentrations for student resumes, reflection papers for the mock interview with an industry professional, the career fair and the job shadow and the concentration portfolio group critiques Next, instructors use a studio-based interdisciplinary approach to assess several of the phases of the electronic portfolio. In particular, the instructors rotate between all the teams to discuss the portfolios of each team. This phase is particularly helpful to the students and the instructors because each instructor has a different expertise and therefore students are able to receive critiques from several different viewpoints.

4.3 External Assessment

External assessment is used in our capstone seminar course in several ways. First, the campus career services office provides feedback to students on their resumes during one class period after which students are then asked to incorporate the suggestions in their resumes. Second, industry professionals are invited to campus to conduct mock interviews with students, which allow students to receive friendly feedback first-hand from an industry professional regarding how they conducted themselves during an interview. Industry professionals have been willing to participate in this exchange because it provides them with an opportunity to search for the best talent but also most say that are willing to assist students develop "soft skills". Third, industry professionals are invited to campus once during the semester to share with students what skills and knowledge they expect from a student who will be seeking their first full-time position. This experience has been beneficial because industry professionals often mirror information that we have been sharing with them about their industry. Fourth, industry professionals provide feedback on the student's portfolio when they job shadow. This allows the students to receive feedback from an industry professional before they seek full-time employment. Many students have received offers for a summer internship or full-time employment from the job shadow and portfolio critique experience and industry professionals are always eager to discover talent for their companies. Finally, the visits from industry professionals provide an opportunity for feedback to the course instructors about the students and our program, which allows the instructors to validate our program and make modifications where needed.

5. COURSE CHANGES, IMPACT OF COURSE, AND CHALLENGES

Our capstone seminar course has changed since 2004 based upon lessons we have learned. First, we have augmented additional opportunities for interaction within the interdisciplinary teams. Comments from industry professionals about the importance of team work within their industries and after witnessing first-hand the benefits of team interactions encouraged us to incorporate more team interactions. Second, we have fine tuned the assessment phases used to develop the electronic portfolio by creating progressive check points throughout the process. In the first year, we incorrectly expected that the students would follow the process we laid out to develop their portfolios, but learned quickly that students are motivated by assessment check points. Third, we now provide a form of team assessment on the electronic portfolio teams. We determined that there are benefits to having students assess the effectiveness of their team members. Finally, we meet prior to the semester to plan the course assignments and activities, regularly during the semester to discuss the progress of students in the course and at the end of the semester to evaluate the effectiveness of the course. We find that good communication between the instructors leads to a successful course and a richer experience for the students.

The impact of the course for students is profound. First, students are provided an opportunity to work with interdisciplinary teams, which often model the particular industry, they will enter. Second, many students receive job offers for internships or full-time employment prior to graduation, which we directly relate to the "soft skills" student have developed, the job shadow with an industry professional and their electronic

portfolio. Finally, faculty and student contact with industry professionals has strengthened our majors and this course. Through contacts from this course, many industry professionals now serve on the various department advisory boards, are willing to participate in the student mock interviews and job shadows. This has provided opportunities for additional feedback from industry on our Interactive Digital Media curriculum.

Challenges for this course have been when students have not completed all the prerequisite core or significant concentration courses. While this does not happen often, it limits the ability of the students to finish the portfolio during the semester. Other challenges have been the pace of the course and the need for the instructors to devote a great deal of time assessing course assignments. While the instructors receive great satisfaction in working with students, the art and mass communication instructors teach this one-hour course as an unpaid overload.

Gonzalez, Cranitch and Jo suggest that the disciplines that may lay claim to multimedia "have traditionally antagonistic cultures" [5, p. 89]. Additionally, they point out that "The lack of cohesion and identity in the context of cross faculty implementation would limit the quality and depth of knowledge that could be imparted" [5, p. 90]. However, our experience has been very different. We created this major with a philosophy that each department brings unique qualities to the major and that no one department may claim the major as their own. We consider our major unique because of the shared governance process. We encourage other universities and colleges to explore a similar interdisciplinary working relationship as they develop new majors.

Our experience has been a pleasant working relationship in which we must compromise with each other to achieve one goal. This concept of one goal is what Klein says separates a multidisciplinary program from a true interdisciplinary program. She notes that at the highest level "is a conscious attempt to integrate materials from various fields of knowledge into 'a new, single, intellectually coherent entity.' This demands an understanding of the epistemologies and methodologies of other disciplines and, in a team effort, requires building a common vocabulary." [7, p. 57]. Two aspects that contribute to the success of our capstone course are the personalities involved and the willingness of faculty to develop knowledge in other domain areas.

6. BROADER APPLICATIONS

The lessons on interdisciplinary capstone courses are that computer science students working on capstone projects would benefit by reaching out beyond their own department to work with other disciplines. Possible collaborations might be working with the department of psychology on evaluating software usability or business programs on the viability of a particular software project.

7. REFERENCES

[1] Bowers, M. Y. & Scherpereel, C. M. (2008). Bizblock: A Cross-Disciplinary Teaching and Learning Experience. Business Communication Quarterly 71(2), p. 221-226.

[2] Boyer, E. (1990). Scholarship Reconsidered: Priorities Professoriate, Carnegie Foundation Special Report. Princeton University Press, New Jersey.

[3] Clear, T., Goldweber, M., Young, F., Leidig, P., Scott, K., (2001). Resources for Instructors of Capstone Courses in Computing, ACM SIGCSE Bulletin. 33(4). Pp. 93-113.

[4] Docherty, M. & Brown, A. (2001): Studio-based teaching in information technology. Flexible Learning for a Flexible Society: Proceedings of the ASET/HERDSA Conference 2000 Joint International Conference 2-5 July, 2000. Toowoomba, Queensland: Australian Society for Educational Technology and Higher Education Research and Development Society of Australasia. Retrieved August, 2008 from http://www.ascilite.org.au/aset-archives/confs/aset-herdsa2000/procs/docherty.html

[5] Gonzalez, R., Crantich, G. & Jo, J. (2000, January). Academic Directions of Multimedia Education. Communications of the ACM. 43(1), 89-95.

[6] Hudhausen, C. D., Narayan, N. H. & Crosby, M. E. (2008). Exploring studio-based instructional models for computing education. SIGCSE Bulletin. 40(1). Pp. 392-296.

[7] Klein, J.T. (1990). Interdisciplinary: History, Theory, & Practice. Wayne State Press, Detroit.

[8] Spradling, C., Strauch, J. & Warner, C. (2008). An Interdisciplinary Major Emphasizing Multimedia. ACM SIGCSE Bulletin. 40(1). pp. 388-391.

[9] Woods, Charlotte. (2007). Researching and developing interdisciplinary teaching: towards a conceptual framework for classroom communication. Higher Education. 54:853-866.

A COMPARATIVE STUDY OF INFORMATION SECURITY AND ETHICS AWARENESS IN DIVERSE UNIVERSITY ENVIRONMENTS*

Max North, DeAnthony Perryman, Shekinah Burns, and Sarah North
Engineering Technology & Management
Southern Polytechnic State University
max@acm.org

ABSTRACT

People involved in management information systems face several challenges, especially when it comes to securing their information systems. This is most evident within the university environment, specifically when providing computing resources to students in diverse university environments. There is a common belief that students who attend technology universities have more awareness of security and ethics than those who do not. This study was conducted to compare the levels of information security and ethics awareness of students in diverse university environments. It compares survey data collected from two different university environments-a liberal arts university and technology university. The result of this comparative study shows that students attending the technology university tend to be more aware of security and ethics in information systems than those who attend the liberal arts university. Finally, based on analysis of the collected data, several recommendations are provided to increase the awareness of computer security and ethics in university environments.

INTRODUCTION

Information security and ethics awareness of information systems plays an important factor in university environments, especially in today's technological era [4]. One pressing challenge of maintaining information systems within university environments

is determining how aware students are of security and ethics, and then providing the appropriate measures to help users and information technology staffs protect these systems [1, 3, 6]. Although there has been data from student surveys relating to awareness and security, there were not any data that related directly to student population of technology universities or to liberal arts universities until the two recent studies conducted by the primary authors of this article [7]. Along with previous work, these recent studies still make it evident that computer security and ethics education is imperative for university users. In a quantitative survey conducted by EDUCAUSE, only a third of higher education institutions offer security and awareness training for students and faculty [2].

The term liberal arts denotes a curriculum that imparts general knowledge and develops the student's rational thought and intellectual capabilities, unlike the professional, vocational, technical curricula emphasizing specialization. Specifically, technology denotes a curriculum that deals with the creation and use of technical means and their interrelation with life, society, and the environment. Technology universities distinguish themselves from liberal arts universities in that they are typically more exposed to information systems because of the amount of related courses and majors offered [12]. Students of liberal arts universities, however, may not be as exposed, due to a lack of related courses and majors [7, 8]. This study is an attempt to compare the data of the information security and ethics awareness surveys conducted in two different university environments, thus identifying what causes students of one university environment to be more aware than others in a different environment.

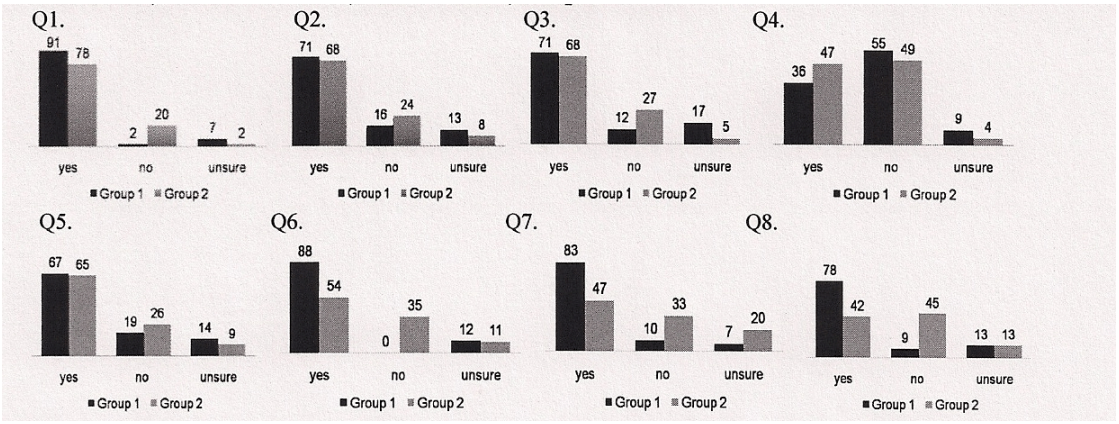
METHODOLOGY

More than one thousand volunteers participated in this study. There were approximately five hundred participants from the liberal university, with the remainder coming from the technology university. The participants responded to a questionnaire, in two major parts, consisting of 24 questions total. The first part of the questionnaire assessed the awareness of computer security. It was comprised of several topic sections: (i) passwords, (ii) data backups, (iii) antivirus software, (iv) firewalls, (v) software updates and patches, and (vi) uninterruptible power supplies. The second part measured the level of ethics awareness in computer use. It was a brief section covering the code of ethics. The questionnaire answers were either yes, no or unsure. The questionnaire was then given quantitative values in the form of percentages. The data were analyzed and compared to identify significant differences, and the results were generated into bar charts. Details of the analyzed data are provided following each section.

RESULTS

Data collected from the two universities were analyzed to determine any possible difference that might occur. For promoting the simplicity of visualization techniques, the universities were depicted in two groups: Group-1 directly pertains to the technology university, and Group-2 refers to the liberal arts university. A z-test statistical technique was performed to determine whether the differences between the universities were significant. The z-test directly compared the percentages at a 99% confidence level.

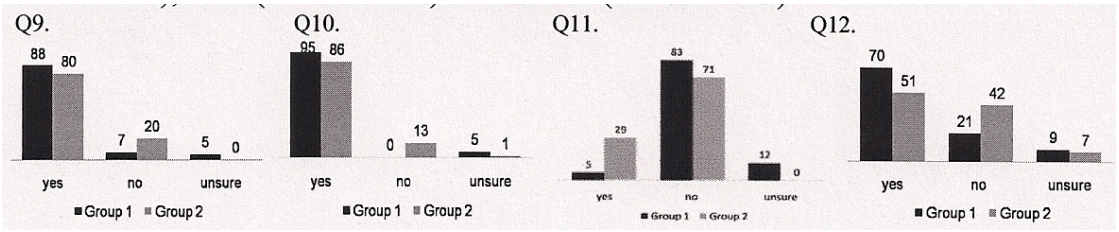
Antivirus Software - Antivirus software is one the most effectively proactive ways to secure, protect, and maintain privacy from unauthorized access to computer resources. Group-1 and group-2 were asked if there was antivirus software installed on the respondent's personal computer. 91% of group-1 answered "yes", while 78% of group-2 answered "yes". 2% of group-1 and 20% of group-2 replied "no". 7% of group-1 and 2% of group-2 answered "unsure". Analysis shows that there is a significant difference between the two groups in those who answered "yes" (z-value=5.59), "no" (z-value=8.99) or "unsure" (z-value=3.66) to question 1. Question 2 asked if the respondent had a current antivirus subscription; 71% of group-1 and 68% of group-2 answered "yes". 16% of group-1 and 24% of group-2 answered "no". 13% of group-1 and 8% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "no" (z-value=3.08), and "unsure" (z-value=2.48) to question 2. There appear to be no significance difference between "yes" responses (z-value=0.96) for both groups. Question 3 asked if the respondent's antivirus software performed automatic updates. 71% of group-1 and 68% of group-2 answered "yes". 12% of group-1 and 27% of group-2 answered "no". 17% of group-1 and 5% of group-2 answered "unsure". There is a significant difference between those who answered "no" (z-value=5.90) and "unsure" (z-value=5.96). There appears to be no significance difference between responses to "yes" (z-value=0.96). Question 4 asked if the respondents performed manual updates to their antivirus program. 36% of group-1 and 47% of group-2 answered "yes". 55% of group-1 and 49% of group-2 answered "no". 9% of group-1 and 4% of group-2 answered "unsure". There is not any significant difference in "no" responses (z-value=1.83) between the two groups. There is a significant difference between the two groups in those who answered "yes" (z-value=3.46), and "unsure" (z-value=3.08) to question 4. Question 5 asked if the participants were aware the university computers have antivirus software. 67% of group-1 and 65% of group-2 answered "yes". 19% of group-1 and 26% of group-2 answered "no". 14% of group-1 and 9% of group-2 answered "unsure". There is not a significant difference in "yes" (z-value=0.60) responses of both groups. There is a significant difference between the two groups in those who answered "no" (z-value=2.57), and "unsure" (z-value=2.38) to question 5.



Firewalls - Firewalls are a type of security measure used to prevent unauthorized access of computer data. Question 6 asked if the respondent knew what a firewall consists of. 88% of group-1 and 54% of group-2 answered "yes". 0% of group-1 and 35% of group-2 answered "no". 12% of group-1 and 11% of group-2 answered "unsure". There is a

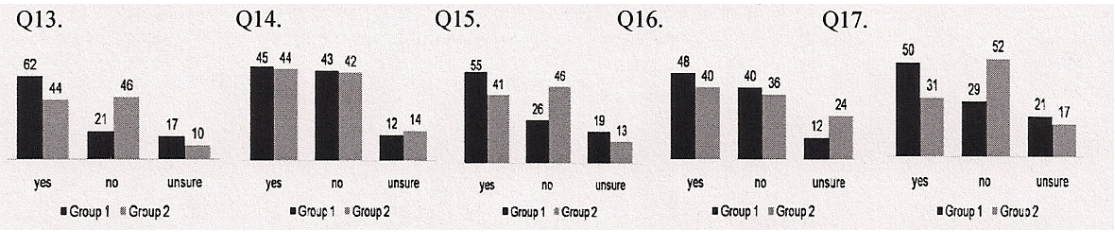
significant difference between the two groups in those who answered "yes" (z-value=11.77) or "no" (z- value=13.91). There appears to be no significance difference between responses of "unsure" (z-value=0.39) for both groups. Question 7 asked if respondents had a firewall installed on their personal computers. 83% of group-1 and 47% of group-2 answered "yes". 10% of group-1 and 33% of group-2 answered "no". 7% of group-1 and 20% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value= 11.86), "no" (z-value=8.77) or "unsure" (z-value=5.92). Question 8 asked if respondents were aware the university has a network firewall. 78% of group-1 and 42% of group-2 answered "yes". 9% of group-1 and 45% of group-2 answered "no". 13% of group-1 and 13% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=11.55) or "no" (z-value=12.75). There is no significant difference in responses of both groups who answered "unsure" (z-value=0.09).

Passwords - Passwords are a set of codes primarily used as a deterrent to unauthorized access. Question 9 asked if the respondent had a password on his or her personal computer. 88% of group-1 and 80% of group-2 answered "yes". 7% of group-1 and 20% of group-2 answered "no". 5% of group-1 and 0% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=3.36), "no" (z-value=5.92) or "unsure" (z-value=4.13). Question 10 asked if respondents had to use a password on university computers. 95% of group-1 and 86% of group-2 answered "yes". 0% of group-1 and 13% of group-2 answered "no". 5% of group-1 and 1% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered 'yes" (z-value=4.74), "no" (z-value=7.68), or "unsure" (z-value=3.52). Question 11 asked if the respondent's password was easy to guess. 5% of group-1 and 29% of group-2 answered "yes". 83% of group-1 and 71% of group-2 answered "no". 12% of group-1 and 0% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=10.01), "no" (z-value=4.43) or "unsure" (z-value=7.31).



Software Updates and Patches - Software updates and patches are an essential part of maintaining proper protection against malware. Question 12 asked if the respondent kept current with software updates and patches against the latest security threats on personal computers. 70% of group-1 and 51% of group-2 answered "yes". 21% of group-1 and 42% of group-2 answered "no". 9% of group-1 and 7% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=6.08), "no" (z-value=3.52) or "unsure" (z-value=3.52). Question 13 asked if respondents were aware that the university keeps current with software updates and patches. 62% of group-1 and 44% of group-2 answered "yes". 21% of group-1 and 46% of group-2 answered "no". 17% of group-1 and 10% of group-2 answered "unsure".

There is a significant difference between the two groups in those who answered "yes" (z-value=5.63), "no" (z-value=8.30) or "unsure" (z-value=3.14).

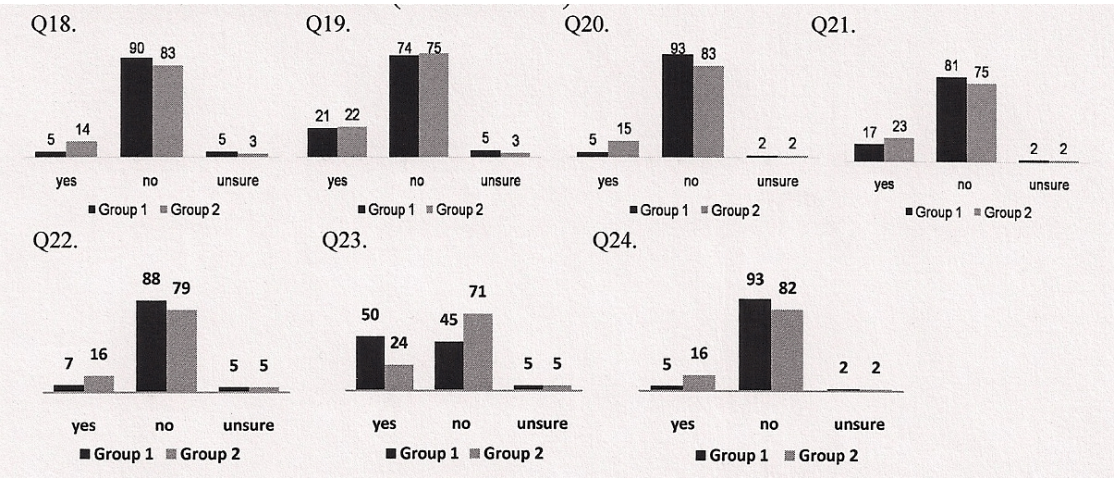


Data Backups - Data backup is the process of copying data to an external source or system. The data is backed up as a preemptive measure to secure its access in case the original data becomes inaccessible. Question 14 asked if the respondent had a current backup of personal data. 45% of group-1 and 44% of group-2 answered "yes". 43% of group-1 and 42% of group-2 answered "no". 12% of group-1 and 14% of group-2 answered "unsure". There is not a significant difference between the two groups in any of the categories; "yes" (z-value=0.25), "no" (z-value=0.25), or "unsure" (z-value=0.84). Question 15 asked if respondents were aware that the university keeps a current backup of personal data. 55% of group-1 and 41% of group-2 answered "yes". 26% of group-1 and 46% of group-2 answered "no". 19% of group-1 and 13% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=4.36), "no" (z-value=6.52), and "unsure" (z-value=2.50).

Uninterruptible Power Supply (UPS) - An Uninterruptible Power Supply is used to protect data from the constant, but rare, threat of power surge or electrical failure. Question 16 asked if the respondent had a UPS to protect data in the event of a power surge. 48% of group-1 and 40% of group-2 answered "yes". 40% of group-1 and 36% of group-2 answered "no". 12% of group-1 and 24% of group-2 answered "unsure". There is a significant difference between the two groups in two categories: "yes" (z-value=2.48), and "unsure" (z-value=4.85). There is no significant difference in responses of both groups to "no" (z-value=1.23). Question 17 asked if the respondent was aware that university has UPSs for protection against power surges. 50% of group-1 and 31% of group-2 answered "yes". 29% of group-1 and 52% of group-2 answered "no". 21% of group-1 and 17% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=6.05) and "no" (z-value=7.34). There is no significant difference in responses of both groups to "unsure" (z-value=1.53).

Code of Ethics - Codes of ethics are the rules set in place to provide universally acceptable guidelines for making ethical decisions [9, 12]. Although the questions provide a broad overview of what ethics are, it is difficult to sum extensive personal beliefs into standardized qualitative data. Qualitative data will have the tendency to be based on cultural beliefs and should be kept with an open mind the data will vary from person to person and group to group. Question 18 asked if the respondent had ever used a computer to harm other people. 5% of group-1 and 14% of group-2 answered "yes". 90% of group-1 and 83% of group-2 answered "no". 5% of group-1 and 3% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=4.74) and "no" (z-value=3.14). There is no significant difference in responses of both groups to "unsure" (z-value=1.45). Question 19 asked if

the respondent ever snooped around in other people's computer files. 21% of group-1 and 22% of group-2 answered "yes". 74% of group-1 and 75% of group-2 answered "no". 5% of group-1 and 3% of group-2 answered "unsure". There is not a significant difference between the two groups' responses in all the categories; "yes" (z-value=0.30), "no" (z-value=0.29) and "unsure' (z-value=1.45). Question 20 asked if the respondent had ever used a computer to steal or bear false witness. 5% of group-1 and 15% of group-2 answered "yes". 93% of group-1 and 83% of group-2 answered "no". 2% of group-1 and 2% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=5.16) and "no" (z-value=4.76). There is no significant difference between the responses of both group to "unsure" (z-value=0.22). Question 21 asked if the respondent had ever used other people's computer resources without authorization. 17% of group-1 and 23% of group-2 answered "yes". 81% of group-1 and 75% of group-2 answered "no". 2% of group-1 and 2% of group-2 answered "unsure". There is not a significant difference between the two groups' responses in any of the categories; "yes" (z-value=2.29), "no" (z-value=2.21) and "unsure" (z-value=0.22). Question 22 asked if the respondent had ever interfered with other people's computer work. 7% of group-1 and 16% of group-2 answered "yes". 88% of group-1 and 79% of group-2 answered "no". 5% of group-1 and 5% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=4.36) and "no" (z-value=3.74). There is no significant difference in "unsure" responses (z-value=0.14). Question 23 asked if the respondent had ever copied or used proprietary software for which he or she had not paid. 50% of group-1 and 24% of group-2 answered "yes". 45% of group-1 and 71% of group-2 answered "no". 5% of group-1 and 5% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=8.44) and "no" (z-value=8.26). There is no significant difference in responses of "unsure" (z-value=0.14). Question 24 asked if the respondent had ever permitted someone else to use your computer/account for illegal purposes. 5% of group-1 and 16% of group-2 answered "yes". 93% of group-1 and 82% of group-2 answered "no". 2% of group-1 and 2% of group-2 answered "unsure". There is a significant difference between the two groups in those who answered "yes" (z-value=5.57) and "no" (z-value=5.16). There is no significant difference in responses of those who answered "unsure" (z-value=0.22).



CONCLUSION AND RECOMMENDATION

There are important and significant differences between understanding of information security and ethical computer use among students in technology universities and liberal universities. The way the individuals are taught and the way the students learn will create a large variance in each culture's perspective. The largest difference in perspective was found in the sections pertaining to firewalls, software updates and patches, uninterruptible power supply, and briefly in the codes of ethics. Members of the technology university had a greater awareness of what a firewall is, as well as its personal and college-based use. The technology students were more aware of software updates and patches to a significant degree. Although neither the technology university nor the liberal university students had an overwhelming awareness of what an uninterruptible power supply is or what it is used for, the technology university was significantly more aware the university campus was using one. In the area of code of ethics, a significant difference presented itself in the questions of using software not paid for or the use of an account for illegal reasons. In reference to the use of software not paid for, the technology college used the software significantly more than liberal student. However, when asked if their account had been used for illegal reasons, the liberal university students said yes significantly more than students of the technology university. Proper measures can be taken to increase awareness among students of both universities [11, 13, 14], including

1. Increased exposure to computers.
2. Detailed training will ensure proper use and maintenance of the computers.
3. Providing the proper tools to the university bodies.
4. Offering classes periodically as technology changes.
5. Developing an initial university computer awareness class to aid in universal knowledge.

ACKNOWLEDGMENT

This effort was supported by the Department of Defense (DOD)/National Security Agency (NSA) Grant. The content of this work does not reflect the position or policy of the DOD/NSA and no official endorsement should be inferred. Southern Polytechnic State University has been designated a Center of Excellence in Information Security Assurance (CAE/ISA) by the Committee on National Security Systems (CNSS) and the National Security Agency (NSA).

REFERENCES

1. Foster, A. (2004). Insecure and Unaware. *Chronicle of Higher Education*, 50(35), A33-A35.

2. Caruso, Judith B. (2003). Information Technology Security: Governance, Strategy, and Practice in Higher Education Key Findings (ID: EKF0305). In *EDUCAUSE*, http://www.educause.edu/ers0305/

3. North. S.M., George, R., Shujaee, K., and Mumford, A. (2005). Collaborative Information Assurance Capacity Building at a Consortium of Colleges and

Universities. *Proceedings of the 43rd Annual Association for Computing Machinery Southeast Conference*, 361-362.

4. Laudon, K.C., Laudon, J.P. (2005). *Essential of Management Information Systems: Managing the Digital Firm* (6th ed.). Prentice Hall.
5. Dinev, Tamara. (2008). Internet Users' Beliefs about Government Surveillance - The Role of Social Awareness and Internet Literacy. *Proceedings from the 41st Hawaii International Conference on System Sciences*, 275-275.
6. North, M., North, S., George, R. (2006). Computer Security and ethics awareness in university environments: A challenge for management of information systems. *ACM SE'06*, 434-439.
7. Myers, J Paul Jr., Riela, Sandra. (2008). Taming the Diversity of information assurance and security. *Consortium for Computing Sciences in Colleges*, 173-179.
8. Holland-Minkely, Amanda M. (2006). Cyberattacks: A lab-based introduction to computer security. *SIGITE'06*, 39-46.
9. Dorantes, C., Hewitt, B., Goles, T. (2006). Ethical Decision-Making in an IT context: The Roles of personal moral philosophies and moral intensity. *Proceedings of the 39th Hawaii International Conference on System Sciences*, 8, 206c-206c.
10. McCoy, Carrie (2004), "You are the key to security": establishing a successful security awareness program. *Proceedings of the 32nd annual ACM SIGUCCS conference on User services*, 346-349.
11. Hentea, Mariana (2005). A perspective on achieving information security awareness. *Issues in informing science and technology*, 169-178.
12. Epstein, R. (2006). An Ethics and Security course for students in computer science and information technology. *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, 233-236.
13. Dark, M., Harter, N., Morales, L., Garcia, M. (2008). An information security ethics education model. *Consortium for computing sciences in college*, 82-88.
14. Katz, Frank H. (2007). Integrating a security awareness program into an information security course. *Journal of Computing Sciences in Colleges*, 23(2), 181-187.

IDENTIFYING GENE REGULATORY NETWORKS USING EVOLUTIONARY ALGORITHMS*

Carl Davidson
Simpson College
701 North C Street Indianola, IA 50125
(515)288-9256
carl.davidson@simpson.edu

ABSTRACT

This paper discusses experiments with genetic algorithms to reconstruct gene regulatory networks represented as Bayesian networks. In our algorithm Bayesian networks are represented by their topological order. Various selection methods and fitness functions were tested on data generated from a small-scale Bayesian network. The algorithm was run on a large fragment of the human transcriptome to demonstrate scalability. Additional improvements on the implementation are proposed for future research.

INTRODUCTION

Gene Regulatory Networks

Under various environmental conditions, genes may produce proteins at different rates, known as *expression levels*. The proteins have various effects upon a cell, some of which may be upon the expression levels of other genes. As a result, a *gene regulatory network* exists which provides feedback to events both inside and outside the cell [2]. Understanding this regulatory network may be important when altering genetic information without adverse consequences.

To elucidate the gene regulatory network, scientists first create what are known as *microarrays*, in which the expression levels of genes are examined under various environmental conditions. Data collected from microarrays are then analyzed by software to produce plausible gene regulatory networks [2]. Existing algorithms represented gene

regulatory networks in a number of ways [1, 2, 5]. Here, several implementation strategies were investigated for one common representation - Bayesian networks.

Bayesian Networks

Bayesian networks are directed acyclic graphs in which nodes represent variables and edges represent relationships between them. Each variable takes on a single value at any time, the combination of which is known as an *event*. Each event may occur with a degree of certainty, perceived by the network as its frequency [6]. In the context of gene regulatory networks, nodes represent genes, which may be underexpressed, overexpressed, or normally expressed. Edges represent interactions between genes, regardless of whether they contribute to underexpression or overexpression.

Because the runtime of certain operations on Bayesian networks increases exponentially with the number of variables, approximation algorithms are typically used to generate large Bayesian networks. Here, only one such optimization algorithm is discussed - a genetic algorithm.

Genetic Algorithms

Genetic Algorithms find favorable solutions for problems among many potential solutions, in a manner inspired by natural selection. The majority of Genetic Algorithms consist of four stages: creation, selection, mating, and mutation [4]. During creation, a series of potential solutions is randomly created. The solutions are collectively known as a "population." During selection, a mating population is created by selecting the best solutions, which are said to have higher "fitness." While mating, a new population is created by combining solutions from the mating population, thus producing new solutions. The combination of solutions is referred to as "crossover." Finally, during mutation a percentage of the population is randomly altered in some way [3, 4].

This process repeats for a desired period. Each individual repetition is known as a "generation." When proper strategies are employed, the section of the population with the best fitness will increase over several generations. The highest fitness in the population should also increase, eventually creating an approximate, yet satisfactory, solution [3].

Genetic algorithms typically require customization to produce faster, better results for specific problems. Each stage of a genetic algorithm offers its own means of customization. During creation, the size of a population, the means of representing solutions, and the means of creating solutions may vary. During selection, the percentage selected, the method of calculating fitness, and the method of selection may vary. Finally, during mating and mutation, methods for manipulating solutions must be specified [3, 4].

Here, various customizations were made to a genetic algorithm that would generate Bayesian representations of gene regulatory networks. A test dataset was randomly generated from a hypothetical Bayesian network, and customizations were rated on how well the Bayesian network was recreated from test data. Finally, the implementation was used on a large scale, practical application - analyzing gene expression in a large segment of the human transcriptome.

IMPLEMENTATION

Representation

Some previously implemented genetic algorithms have represented Bayesian networks as 2D Boolean arrays, in which each cell signifies whether a relationship exists between variables [1, 5]. Variables are assigned rows and columns in these arrays, which remain constant throughout all Bayesian networks created by the algorithm. One such popular implementation is found in WEKA, or the Wakaito Environment of Knowledge Analysis [1].

Because a 2D array may also represent cyclical graphs, representing acyclic graphs is problematic. Without implementing additional code to validate solutions, Bayesian networks may be generated that violate their acyclic nature. However, implementing validation code jeopardizes efficiency, which is important when dealing with large datasets such as those found in microarrays. Furthermore, because less than half of each 2D array could represent the edges of an acyclic graph, a large percentage of each 2D array would be wasted, again jeopardizing efficiency.

Here we describe an efficient method used to represent valid Bayesian networks in a genetic algorithm. In addition to relationships between variables, each Bayesian network would store their topological order. To conserve memory, relationships could then be represented as a 2D jagged array - a list of arrays of different lengths. Relationships that violate topological order are simply unrepresented.

The topological order of each Bayesian network is represented as a 1D array with a length equal to the number of variables. Each variable is assigned one cell in this array, which contains the position of the variable in the topological order. Variables are then assigned rows and columns in the jagged array, which correspond to their positions in the topological order. Because relationships that violate the topological order are unrepresented, the size of a row at column N in the jagged array equals N. Figure 1 illustrates this design.

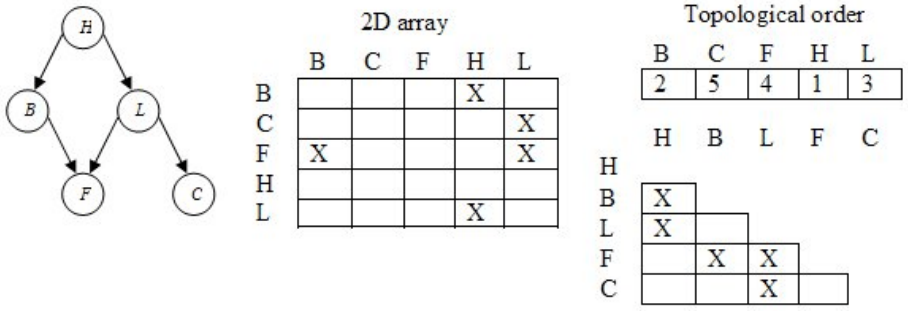


Figure 1. A Bayesian network (*leftmost*) represented using two different methods - a regular 2D array, and a jagged array.

Fitness

All fitness functions examined were based on the entropy metric scoring function implemented by WEKA. Two modifications of the entropy metric were implemented as scoring functions: the AIC metric, and MDL metric [1]. These metrics account for the correlation between values of variables and their parents. For instance, if a gene and its

parents are consistently overexpressed in the sample data, the network's score would be high, meaning that an existing relationship has been identified.

Manipulation

The crossover and mutation of Bayesian networks center upon topological order. Because topological orders are permutations of a single list of numbers, the implementation borrowed heavily from genetic algorithms used to solve the Traveling Salesman Problem. Such genetic algorithms represent potential solutions as permutations of a list of towns, which must be visited exactly once in the shortest possible route [3, 4]. In our implementation, potential solutions are represented as permutations of a list of genes, corresponding to the topological order of the Bayesian network.

Unlike the genetic algorithm used to solve the Traveling Salesman Problem, the relationships of a Bayesian network must be updated to reflect changes in its topological order. In the current implementation, the new network is simply populated by all relationships from the old network that conform to the new topological order. Because a network with n nodes contains $O(n^2)$ possible relationships, this method is of the runtime $O(n^2)$.

Testing

Tests were performed on two datasets. The first dataset was a collection of 100 samples randomly generated using probabilities provided by a trivial Bayesian network, seen in Figure 1. Using this dataset, the best network of each generation was recorded for 20 generations. This was repeated 50 times for each fitness and selection function implemented. Selection functions included tournament, rank-based, and elitist selection methods. Rank based selection was tested using both 2 and 10 ranks. Fitness functions included the entropy, AIC, and MDL metrics. The best networks ever generated during tests were then compared with the network described in the dataset.

The second dataset was a fragment of a microarray performed on the complete human transcriptome. Half of the 158 samples and a quarter of the 22,216 genes were included in the dataset, which was the largest dataset possible on our available hardware. Numerous experiments were performed to evolve populations of 50 networks over several thousand generations, which was, again, the largest number possible on our available hardware. Observations were made on the runtime and on the results of the genetic algorithm while processing this dataset.

RESULTS

Best solutions were consistently higher on the first dataset when forms of rank based selection were employed. 10-Rank selection typically reached its peak fitness in a shorter number of generations. However, 2-Rank selection generated slightly better solutions at a steady rate. As a result, it is believed analysis using 2-Rank selection is less likely to return solutions at a local optimum. Figure 2 illustrates these findings. All fitness scores

are relative to the dataset used to generate networks - that is, networks from different data sets cannot be compared through their scores.

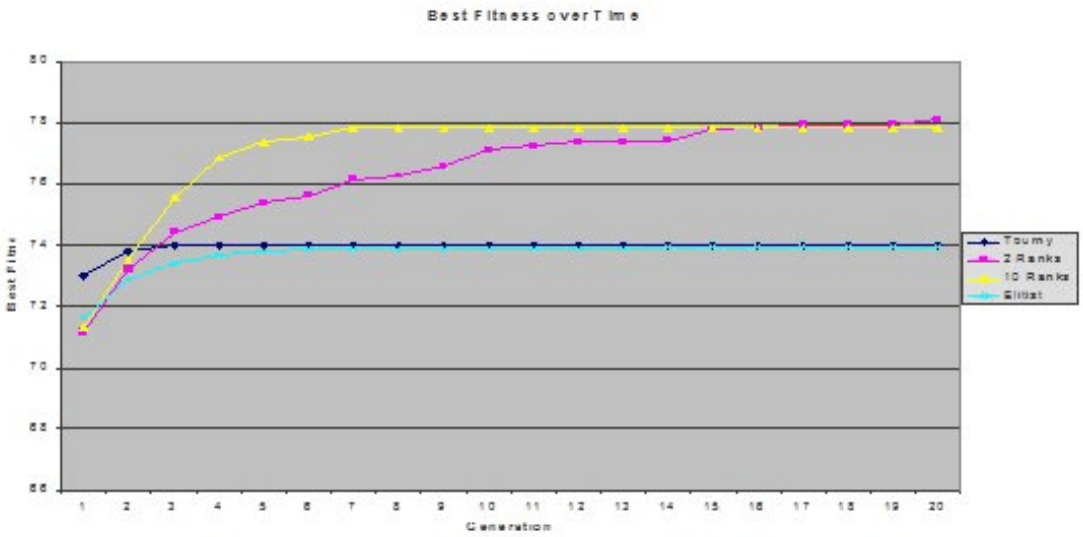


Figure 2. Fitness of best solutions over time, using four selection functions.

The 2-Rank selection method and the MDL metric were employed to analyze the fragment of the human transcriptome microarray. Each network consumed approximately 17.5 MB storage, and an average of 5 seconds to generate and analyze on our available hardware. Resource consumption was greatest during selection and mating operations, when duplicates of the population were created to which existing networks were imported.

Each network generated had an MDL fitness score ranging from approximately 40 000 to 1.4 x 10^7. Curiously, the fitness of networks fell within ranges, those being 40 000 to 60 000, 1 x 10^6 to 2 x 10^6, near 5 x 10^6, and near 1.4 x 10^7. Only two iterations of the genetic algorithm ever produced networks with MDL fitness scores within the 1.4 x 10^7 category. Each iteration did so within the initial generation.

Each network generated 0 to 7 relationships per gene, with an average of 2 relationships per gene. This number likely reflects the probability at which relationships are randomly generated in the initial population, rather than the actual average of relations per gene.

DISCUSSION

Best solutions were consistently closer to the original Bayesian network when the MDL metric was used as a fitness function. Best solutions generated using AIC and entropy metrics were often characterized as having one variable with relations to most other variables, while the other variables had little to no other relations. No reason for this behavior was found.

No network was generated that flawlessly reconstructed the Bayesian network expressed in test data. However, the majority of the best networks generated shared

similar topological orders with the actual Bayesian network. Figure 3 illustrates the 3 best networks generated.

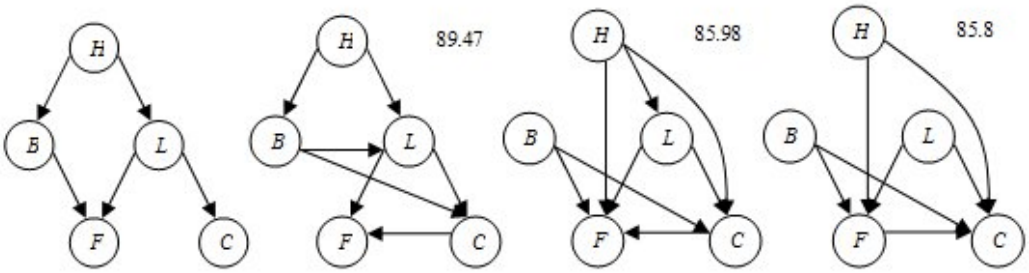


Figure 3. A Bayesian network (*leftmost*) and the best reconstructions generated. Numbers next to each graph denote fitness using the MDL fitness function.

It is important to note the genetic algorithm could reconstruct only the topological order expressed in test data. Because the topological order allows much less variation than the relationships themselves, it is possible the algorithm was only capable of finding the topological order given inadequate time or data. However, because the topological order was used to represent networks, it is possible this is an inherent characteristic of the algorithm. In either case, additional generations or larger populations may still alleviate the issue - it is still possible for the algorithm to generate any valid Bayesian network, and providing additional generations and larger populations increases the likelihood of the algorithm to generate the correct one. However, this is a very resource intensive solution. A more efficient solution would use the algorithm to determine the topological order of the Bayesian network expressed in the data, and then use existing algorithms to determine the specific relationships of the Bayesian network, given the topological order.

Analysis of a portion of the human transcriptome demonstrated the algorithm could scale to large, real world applications. However, the sheer size of the networks made it difficult to determine any visible patterns among their relationships, if any existed. Based on results using test data, it is presumed best solutions could reliably predict only the topological order of such regulatory networks, and patterns among relationships would be limited. Furthermore, the average number of relations per gene was close to the default value specified, and it is presumed not enough generations were simulated to produce reliable relationships that would occur in patterns. For these reasons, extracting any practical information on gene regulation is unsound, for the moment.

CONCLUSION

Understanding gene regulatory networks contributes to knowledge in genetics, and in turn, quality of life. Here, a genetic algorithm was implemented to elucidate gene regulatory networks using Bayesian networks. Various selection methods, and fitness functions were examined based upon efficacy. A method for representing Bayesian networks in genetic algorithms was also implemented, which stores the topological orders of each network to improve efficiency. The implementation was then tested on a real life microarray. Results demonstrated the method was able to reconstruct equivalent

topological orderings of the sample network, but could not completely reconstruct the network's relations. Rank based selection and the MDL fitness scores appeared to generate the best solutions. Some features of the microarray could be extracted from the results, but the size of these results prevents manual interpretation.

ABOUT THE PAPER

This project was performed over the 2008-2009 school year by an undergraduate student of Simpson College, Iowa. Its subject matter was proposed by Dr. Lydia Sinapova to fit the student's interests in Computer and Biological Sciences. Most all other aspects of the work were performed by the student. In the context of CS Education, it demonstrates the possibilities for high quality, interdisciplinary undergraduate research projects where an interest exists.

The project was sponsored by the Iowa College Foundation through the Maytag-McElroy Undergraduate Research Grant. Many thanks to Dr. Sinapova for her support.

REFERENCES

[1] Bouckaert, R., Bayesian network classifiers in Weka, weka.sourceforge.net/manuals/, retrieved 2/1/2009.

[2] Do, K., et al. *Bayesian Inference for Gene Expression and Proteomics*, New York: Cambridge University Press, 2006.

[3] Goldberg, D., *Genetic Algorithms in Search, Optimization & Machine Learning*, Boston: Addison Wesley Longman, Inc, 1989.

[4] Haupt, R., Haupt, S., *Practical Genetic Algorithms*, New York: John Wiley & Sons, Inc, 1998.

[5] Larranaga, P., et al. Structure learning of Bayesian networks by genetic algorithms, *Pattern Analysis and Machine Intelligence*, 18, (9), 912-926, 1996.

[6] Neapolitan, R., *Learning Bayesian Networks*, Upper Saddle River, NJ: Pearson Education, Inc, 2004.

JAVA SLOT MACHINE APPLET*

NIFTY ASSIGNMENT

Thomas Mertz
Department of Engineering Technology
Kansas State University at Salina
2310 Centennial Road
Salina, KS 67401
(785) 826-2602
tmertz@ksu.edu

ABSTRACT

This paper describes an assignment in which a student implements a Java applet that simulates a slot machine. The assignment is given in an advanced Java programming course in conjunction with the topic of multithreading. The assignment's student learning objectives include: (1) implement multiple concurrent threads, (2) recognize and use the Java `synchronized` construct, (3) recognize and use thread states to pause and resume thread execution, (4) use nested panels to construct an attractive GUI. Because most students know what a slot machine is, they immediately understand the requirements of the assignment and most think it is fun.

THE COURSE

The assignment is used in a three-credit course covering advanced Java programming. Students taking this course have already completed three prerequisite computer courses (nine credits). These are a 3-credit course in algorithm development, a 3-credit course in computer hardware and architecture and a 3-credit Java programming course that covers material roughly equivalent to the first eleven chapters of Deitel and Deitel [1].

The advanced Java programming course covers a sequence of topics similar to that of the subsequent chapters of Deitel and Deitel.

THE ASSIGNMENT

For this assignment, each student implements a Java applet that simulates a slot machine. The applet presents the user with three spinners and a play button. When the user clicks the play button, each of the spinners begins to cycle through a sequence of images. Each spinner has its own stop button. When the user clicks a stop button, the associated spinner stops on the most recently displayed image. The applet has a "stomp" button that, when clicked, stops all spinners. Figure 1 shows a user interface for the slot machine designed by a student. It is fairly typical of what students come up with.

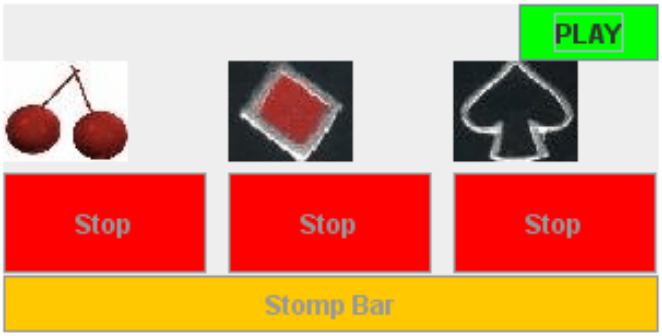


Figure 1 – Typical Slot Machine User Interface

STUDENT LEARNING OBJECTIVES

The primary purpose of the assignment is to cover the topic of multithreading. Students must implement each spinner as a concurrent thread. The play and stop buttons require the spinners to pause and resume execution so the students must understand thread states and be able to use the **`synchronized`** construct to implement the thread states of running and waiting.

Prior to this assignment, students have already completed chapters and assignments on applets and GUI programming. For the topic of multithreading, I introduce the students to material similar to the first three sections of the concurrency lesson in the Java Tutorials [3]. This includes basic concepts of concurrency and how to implement a concurrent thread using the **`Thread`** class in the Java API [2]. I follow this with an introduction to thread states similar to that in Deitel and Deitel [1] (pages 1059-1067). I find it useful to provide an example containing a thread that can be paused and resumed using stop and go buttons as this is a topic that cannot be readily found in references.

Since the primary focus of the assignment is on multithreading and not simulation, the students are not required to accurately simulate the workings of a real slot machine or its stochastic behavior.

CONCLUSION

Advantages of the assignment are that it is a fairly straightforward implementation of multithreading. Students need not worry about race conditions as there is no access to common data. Furthermore, programming the play and stop buttons is a useful skill that can be used in subsequent programs and is not covered in most textbooks.

REFERENCES

[1] Deitel, H. M. and Deitel, P. J. *Java, How to Program*. Eighth Edition. Upper Saddle River, NJ: Prentice Hall, 2010.

[2] Sun Microsystems. "Class Thread." *Java™ Platform Standard Edition 6*. 2008. Web. 5 December 2009. <http://java.sun.com/javase/6/docs/api/>.

[3] Sun Microsystems. "Lesson: Concurrency." *The Java™ Tutorials*. 1995-2009. Web. 5 December 2009. <http://java.sun.com/docs/books/tutorial/essential/concurrency/index.html>.

HUMAN ROBOT ASSIGNMENT FOR CS0 OR CS1*

NIFTY ASSIGNMENT

Ernie Giangrande Jr.
Mathematics & Computer Science Dept.
Wingate University
Campus Box 3007
Wingate NC 28174
(704) 233-8000
egiangra@wingate.edu

The purpose of this assignment is (a) to get students familiar with the level of detail required in developing computer programs, and (b) to reduce student anxiety about computer programming, and about the course. This assignment is to write a ‘program' to move a human ‘robot' from one location on campus to some other location on campus. The assignment consists of four phases. Phase One consists of talking about what kinds of instructions might be required to program a human robot and developing a command set for the human robot (see below). Phase Two consists of each student writing a program to move the human robot from the front of the classroom to some specified location on campus. In Phase Three students pair up with student #1 playing the role of a human robot, and following the instructions in student #2's program. Student #2 observes and notes ‘bugs' in his or her program. The pairs of students then return to the classroom and exchange roles. Finally, in Phase Four, each student then takes his or her program, as well as their observations re bugs, and rewrites the program to correct those bugs.

THE ASSIGNMENT:

This assignment is to write a ‘program' to move a human ‘robot' from one location on campus to some other location on campus.

An Example Control Set:

Sequential Constructs

- STEP FORWARD
- STEP BACKWARD
- TURN LEFT d*
- TURN RIGHT d*
- STEP UP
- STEP DOWN
- OPEN DOOR
- WAIT s

Control Structures

- IF/THEN – END IF
- IF/THEN/ELSE – END IF
- WHILE – END WHILE

* where d is the number of degrees, and s is seconds

Conditions

- AT DOOR
- DOOR OPEN
- AT WALL
- AT TOP OF STAIRS
- AT BOTTOM OF STAIRS
- AT PATH INTERSECTION
- AT STREET CROSSING
- CAR COMING

Examples

```
IF (AT WALL) THEN STEP
BACKWARD END IF

IF (DOOR OPEN) THEN STEP
   FORWARD
ELSE
  OPEN DOOR STEP
FORWARD END IF
WHILE (NOT AT DOOR) STEP
  FORWARD
END WHILE
```

APPLIED PROBLEM SOLVING*

NIFTY ASSIGNMENT

Wen-Jung Hsin and John Cigas
Information and Computer Science
Park University
8700 N.W. River Park Drive
Parkville, MO 64152
wen.hsin@park.edu

In a Discrete Math course, to allow students to be creative, think about what they have learned, and be able to apply what they have learned, the authors have created the following assignment:

Assignment:

Applied Problem Solving – Apply a problem solving technique to a life example.

Each student proposes a problem using the problem solving technique studied in class, but in a different context from the scenarios shown in the text. Students post their problems to an electronic discussion board for other students to view and solve.

This is a weekly assignment. The student's choice of the problem domain is based on the weekly reading. After students understand a concept, they need to know how to apply it.

This assignment gives students power and freedom, and many students respond well. Since all student-proposed problems are posted on the electronic discussion board, everyone in class can see what other students have done with the same concept learned in class. This assignment is a very nice reinforcement for concepts that students have learned.

This assignment serves well if the course is accompanied by electronic discussion board for group discussion. This is better done for discussion outside of the classroom because it gives students time to think about the problems and the solutions. However, in-classroom discussion can also be conducted once students have a chance to think about the assignment outside the classroom.

In our presentation, we will show several problems proposed by the students and give summaries of the ensuing discussions.

ANIMATIONS FOR COMPUTER NETWORKING PROTOCOLS*

Wen-Jung Hsin
Associate Professor
Information and Computer Science
Park University
wen.hsin@park.edu

ABSTRACT

This paper introduces several collections of computer networking protocol animations that can be used in conjunction with classroom instruction. Some of these animations allow users to change parameters and settings, thereby enhancing the interactive learning. Additionally, this paper discusses in detail two computer networking protocol animations that were written by students at Park University. These animations have been made publicly available in the student resource website [2] of Kurose and Ross' Computer Networking book [1] and have been used around the globe for several years.

1. INTRODUCTION

The theory of computer networking is the foundation of the Internet and its popular applications such as web, email, and streaming multimedia. However, teaching the theory of computer networking to students without visual aid can be rather dry and uninteresting. Drawing pictures on the board or on a presentation slide is fine; however, drawing takes time and usually misses the live interaction in the network communication. In recent years, various technologies and software can help animate the networking theories. Furthermore, it is a belief that visualization technology has a positive impact on student learning [3]. This paper provides a list of computer networking animation resources that can help teachers teaching the networking concepts. In addition, this paper presents computer networking animations that were written by Park University students to help learners understand how the underlying protocols work.

This paper is organized as follows. Section 2 provides a list of computer networking animations resources. Section 3 presents the Selective Repeat protocol animation. Section

4 presents the Flow Control protocol animation. Finally, section 5 provides a summary and conclusion.

2. LITERATURE RESOURCES IN NETWORKING ANIMATIONS

This section provides and discusses several collections of networking protocol animations in the literature.

Kurose and Ross [1] provide a publicly available collection of Java applets [2] illustrating HTTP, DNS, CSMA/CA, CSMA/CD, Flow Control, Go-Back-N, Selective Repeat, IP fragmentation, packet vs. message switching, transmission vs. propagation delay, queuing and loss, etc. This collection of animations makes the theory more understandable as the students get to explore various networking scenarios by experimenting with different parameters and settings. Park University students have contributed 2 Java applets to this collection of networking animations. In Sections 3 and 4, we will discuss the animations by Park University students in more detail.

Forouzan [4] also provides a publicly available collection of Macromedia Flash animations in [5]. This collection of animations animates the figures in the networking textbook [4]. Unlike [2], the animation scenarios in [5] are static such that users are not allowed to change parameters or settings. Nonetheless, it is still useful in illustrating how some of the network protocols work.

Holliday [6, 7, 8] discusses Java applets featuring a protocol layer stack, error control, and Ethernet.

For network security related animations, Yuan, et.al. [9] introduced an animated simulator for packet sniffer. This animation is written in Macromedia Flash MX. It can be run in a web page as a Flash Applet, or as a standalone application. Yuan, et.al. [10] provided a visualization tool with 5 demonstrations for wireless network attacks. Yuan, et. Al. [11] provided an animated tool for Kerberos authentication. Schweitzer and Baird [12] introduced an interactive visualization applet for teaching ciphers.

Turner and Robin [13] introduced JASPER (Java Simulation of Protocols for Education and Research) as a protocol simulator. Note that [13] is included in [14], a related literature source that gathers the teaching computer networking tools around the globe.

3. SELECTIVE REPEAT PROTOCOL ANIMATION

Selective Repeat (SR) protocol is a protocol that can be used in the Transport layer for reliable data transfer. Park University student Joshua McKinzie wrote the SR protocol animation during 2005-2006 academic year as a class project assigned by the author in a Computer Networking course. This animation has been published in [2], and used world-wide ever since according to the authors of the textbook. Additionally, this animation is linked from the Wikipedia External Resource [15].

In the SR protocol, a receiver acknowledges every packet that it receives correctly, and buffers any packet arriving out-of-order. In this way, the sender only needs to re-transmit the packet that is not acknowledged. This is in contrast with the Go-Back-N

(GBN) protocol in which if a receiver is waiting for the next sequential numbered packet, but when other higher numbered packets arrive, these out-of-order packets are discarded and the receiver sends an acknowledgement for the most recently received in-order packet. Thus, in GBN, an acknowledgement of a packet number *n* indicates all packets up to and include *n* are received correctly by the receiver.

In comparing SR and GBN protocols, GBN may incur a large number of re-transmissions due to a packet error (because out-of-order packets are not being buffered); whereas, SR, trying to solve GBN's problem, requires acknowledging every packet it receives.

As an exercise to illustrate this difference between SR and GBN protocols, the students of a computer networking course at Park University are directed to send 5 packets using both SR and GBN animations in [2]. After 5 packets are sent, students are to kill packet 2, meaning that packet 2 is lost in the transit. Figures 1 and 2 illustrate this scenario. In Figure 1, for SR protocol animation, the acknowledgement for packets 3 and 4 are marked accordingly. After the agreed-upon timeout, the sender knows to re-transmit packet 2 only. In Figure 2, for GBN protocol animation, the acknowledgement for packets 3 and 4 are marked 1, indicating the receiver has received up to packet 1 correctly. Thus, in GBN, after the agreed-upon timeout, the sender will not only need to re-transmit packet 2, but also packets 3 and 4.

After a couple of other experiments (such as killing an acknowledgement) using these animations, the students in general gain a better understanding the difference between SR and GBN protocols.

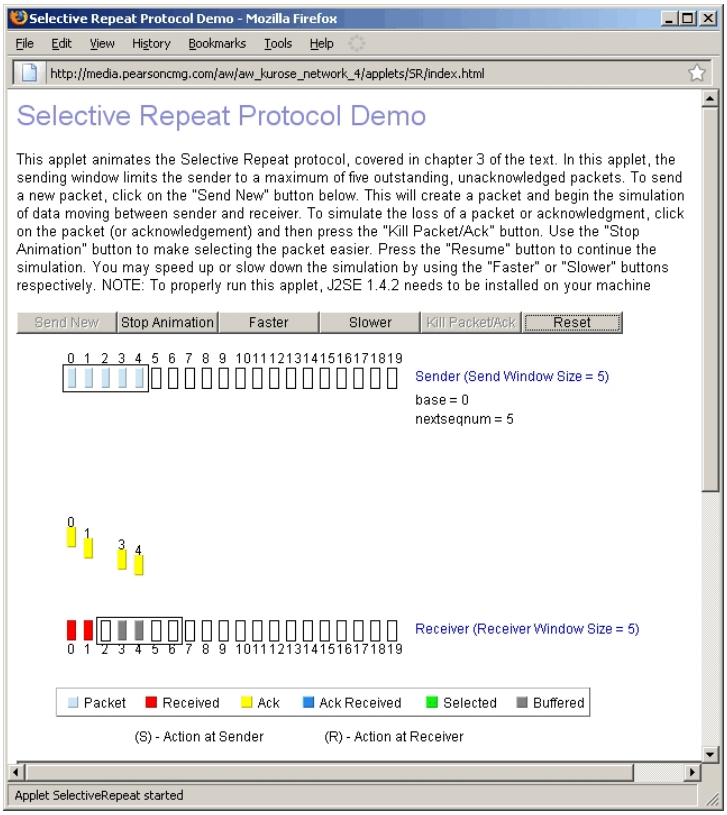


Figure 1. Selective Protocol Animation

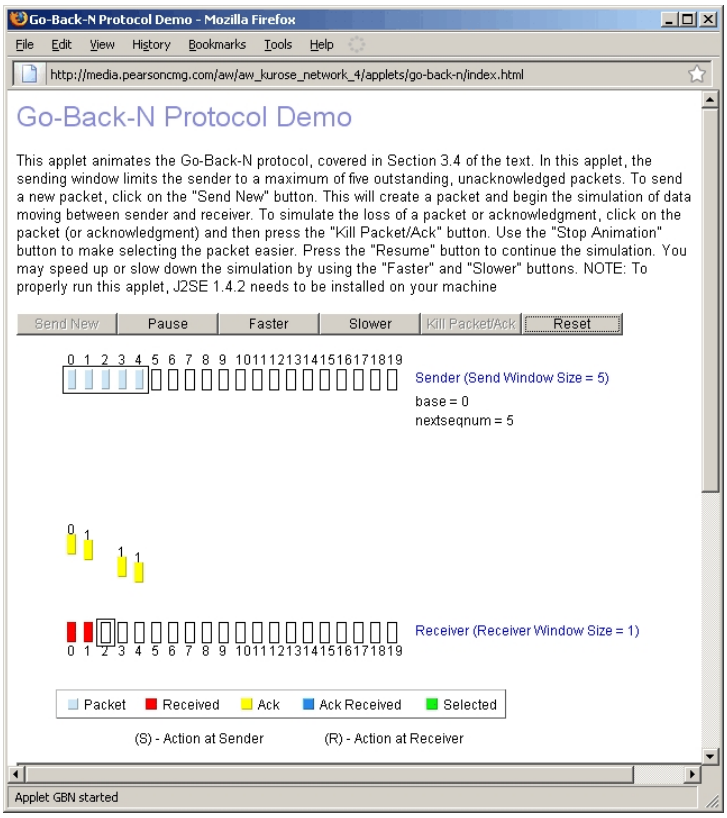


Figure 2. Go-Back-N Protocol Animation

4. FLOW CONTROL PROTOCOL ANIMATION

Flow Control is a mechanism adopted by the Transmission Control Protocol (TCP) in the Transport Layer so that a sender does not overwhelm the receiver's buffer space. In the collection of networking animations [2], the Flow Control animation was originally coded by Hyojin Kim at the University of Pennsylvania in 1997. However, this original animation has several problems, including reporting incorrect window sizes in the acknowledgements. Park University student Rodrigo Neri re-coded the program to fix the problems and enhanced the program (including slower and faster simulation) that is now published in [2].

In the flow control of a TCP connection, the receiver informs the sender how much buffer space is left so that the sender knows how much it can send.

As an exercise, the students of a computer networking course at Park University are directed to set the file size of 8 Kbytes and buffer size of 2 Kbytes as an experiment. Figure 3 illustrates a scenario in which the receiver has successfully received 4096 bytes from the sender and has sent those data to the application layer. Thus, the receiver informs the sender of its buffer space (i.e., WIN = 2048) and ACK = 4096 to the sender.

By observing the headers of packets and acknowledgements in the experiments of this animation, the students, in general, gain a better understanding of how TCP prevents the overflow of the receiver's buffer.

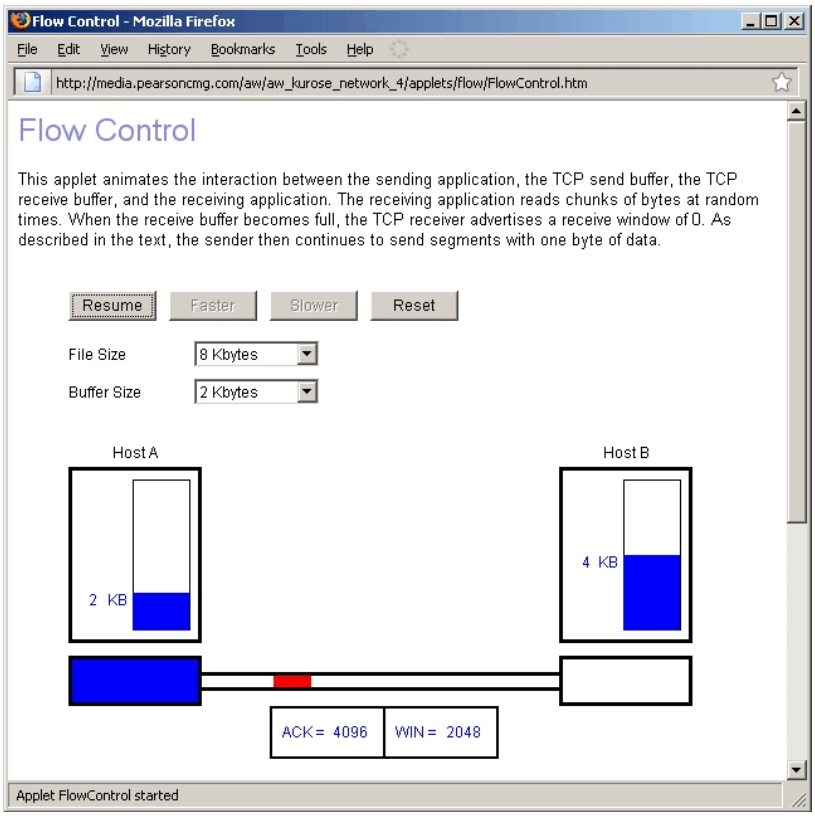


Figure 3. Flow Control Animation

5. SUMMARY AND CONCLUSION

This paper presents collections of networking animations for illustrating how various elements in the networks communicate. Many of these animations are readily available to the instructors and students. With the networking animations, the learners can develop a better understanding of theoretical concepts. It is particularly useful when the animations allow for changing of the input parameters or settings.

REFERENCES

[1] Kurose, J. and Ross, K. Computer Networking, a Top-Down approach featuring the Internet, 5rd edition, Addison Wesley, 2009.

[2] Kurose, J. and Ross, K. Student Resource Java Applet web site, http://wps.aw.com/aw_kurose_network_5/111/28536/7305312.cw/index.html

[3] Naps, T. et.al. "Exploring the role of visualization and engagement in computer science education", ACM SIGCSE Bulletin, vol. 35. Issue 2, pp. 131-152, 2003.

[4] Forouzan, B. Data Communications and Networking, 4th edition, McGraw Hill, 2007.

[5] Forouzan, B. Java Applet Networking animations. http://highered.mcgraw-hill.com/sites/0072967757/student_view0/index.html ➜ Student edition ➜ choose a chapter ➜ More resources ➜ Animations

[6] Holliday, M. "Animation of Computer Networking Concepts", ACM Journal of Educational Resources in Computing, volume 3, issue 2 (June 2003), pp. 1-26 (issue appeared in May 2004). http://polaris.cs.wcu.edu/~holliday/papers/jeric03Paper.pdf.

[7] Holliday, M. and Johnson, M. "A web-based introduction to computer networks for non-majors - the protocol stack", February, 2004. http://cs.wcu.edu/~holliday/cware/Stack/indexStack.html.

[8] Holliday, M. "A Java applet for illustrating Internet error control", Mathematics and Computer Educaiton, Fall 2004, vol. 38, no. 3. Pp. 326-332.

[9] Yuan, X., Vega, P., Xu, J., Yu, H., and Providence, S., "An animated simulator for packet sniffer", Proceedings of WECS7 - The seventh workshop on Education in Computer Security, January 4-7, 2006, Monterey, California.

[10] Yuan, X., Archer, R., Xu, J., and Yu, H., "A visualization tool for wireless network attacks". http://cisr.nps.edu/downloads/wecs7_ch9.pdf.

[11] Yuan, X., Qadah, Y., Xu, J., Yu, H., Archer, R, and Chu, B. "An animated learning tool for Kerberos authentication architecture", Journal of Computing Sciences in Colleges, April, 2007.

[12] Schweitzer, D. and Baird, L. "The design and use of interactive visualization applets for teaching ciphers", Proceedings of 2006 IEEE Workshop on Information Assurance, 2006.

[13] Turner, K. and Robin, I. "An interactive visual protocol simulator", Computer Standards & Interfaces, 23, pp. 279-310. 2001.

[14] Sarkar, N. (Editor). Tools for Teaching Computer Networking and Hardware Concepts, Information Science Publishing, 2006.

[15] Selective Repeat ARQ. Wikipedia. External Link resource. http://en.wikipedia.org/wiki/Selective_repeat

EXPERIENCES WITH ONLINE SQL ENVIRONMENTS*

John Cigas and Barbara Kushan
Computer Science, Information Systems, and Mathematics Department
Park University
Parkville, MO
816-584-6435
john.cigas@park.edu, bkushan@park.edu

ABSTRACT

This paper describes our experiences with teaching SQL using three different online SQL environments: SQLzoo.net, MySQL Query Browser, and Teradata SQL Assistant/Web Edition. Each has been used in formal classes for learning SQL. This paper documents the features of each, along with highlighting the strengths and limitations.

INTRODUCTION

For their first introduction to SQL, we have tried to give students access to preconfigured environments that do not require the setup and loading of a database, thus allowing them to focus on writing queries. One way of doing this is to distribute a single database file for a personal database, like Microsoft Access or OpenOffice Base. While seemingly convenient, this has proven to be less than successful for teaching SQL. Our experience is that students using MS Access tend to use Query By Example (QBE) to build their queries, and then use an SQL view to get the query for grading submission. While this is easily detected and discouraged, it is still common for students to take this path and avoid learning the details of SQL queries.

MS Access also has caused issues with our students taking classes online. Invariably, someone does not have the school's current version, and struggles to work with their database. Keeping multiple file versions, each with its own set of documentation is neither practical nor desirable.

Our strategy then has been to provide tools that can be used with an online server, accessible from anywhere on the Internet. This serves students in all modes of instruction with a single, well-defined environment.

The remainder of this paper outlines the three different SQL environments that we have used SQLzoo.net, MySQL Query Browser, and Teradata SQL Assistant/Web Edition, emphasizing the usefulness and limitations of each one.

SQLZOO.NET

Overview

The simplest and easiest SQL environment to setup and use is SQLzoo.net. This web site, designed and maintained by Andrew Cumming [1], provides brief SQL tutorials, a small variety of non-trivial databases, and opportunities for students to try numerous SQL queries, with automatic feedback for less than perfect attempts. There is absolutely no set-up for the instructor and no registration for the student. All that is required is a web browser along with an Internet connection.

A sample from the web site is shown in Figure 1. Each exercise page gives a description of the table(s) being queried, any supplementary information, an input box for entering a query, and a results box for the query output.

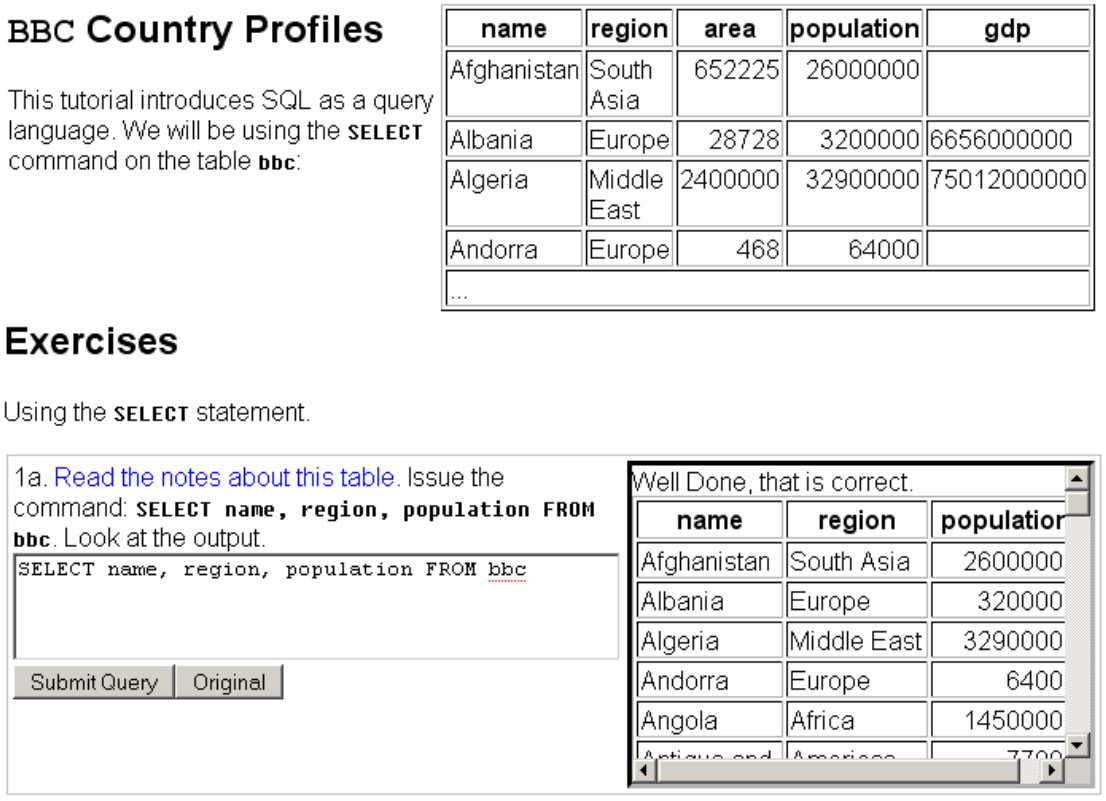


Figure 1

When the user enters a query that generates incorrect data, there is often appropriate feedback, such as that shown in Figure 2.

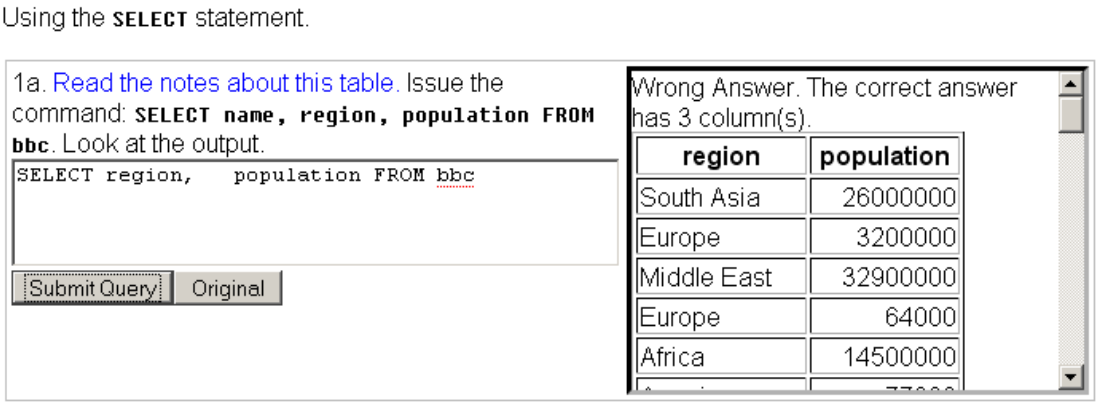


Figure 2

SQLzoo.net provides numerous opportunities to practice simple queries, as well as subqueries, aggregate functions and joins. For more advanced students (or those more curious) there is a way to select different back-end database servers to see the variations in acceptable SQL syntax.

Strengths

The primary benefits of SQLzoo.net are the universal Internet access, complete lack of set up, and a variety of non-trivial databases. Students can get started on these tutorials and exercises with only the briefest of introductions.

The downsides to the tool are the lack of easy way of recording student work. As a tutorial for students to work through on their own, it is outstanding. As an assignment that requires assessment, there are a few problems.

Limitations

Although this is a good tool for tutorials and practice, it can't be used easily for assessed exercises. In keeping with the tutorial spirit, many answers are already posted on the web site. However, even for those tasks meant as assessments, there is no way for students to save their work and resume later from a different computer. There is neither a login nor personal storage on the server. Furthermore, there is no easy way for students to submit their work for assessment. One way is to have students cut and paste the query and then the results from the browser window into a word processing document. While not excruciatingly tedious, it is an extra step that has resulted in errors and inconsistencies. Having students save the HTML page has not been universally successful either. Depending on the browser and browser settings, only some of the queries and results may get saved on the local computer.

In order to provide good feedback, each query of the tutorial is fixed. While it is possible to submit other queries against the database in a provided input box, the corresponding feedback will not reflect the modified assignment and can be a big distraction, especially when it contradicts a set of correct outputs.

No other databases are available. While there are a good variety of different databases, it is not possible to use this site with a database from a textbook.

With a few simple exceptions, this is a tutorial on using the SELECT statement. There are brief examples using other SQL commands (INSERT, UPDATE, DROP, CREATE, etc) but these don't have the same level of interactivity as for writing select statements, where students get to experiment.

The page organization has gotten a little out of kilter. Pages don't flow in numeric order and links don't necessarily take you to the expected next page. For example, Tutorial 1 is on page 1.html, but tutorial 2a is on page 1b.html and tutorial 2b is on page 1a.html!?! This is minor overall, but does require some attention to detail and perhaps more specific instructions from the instructor before sending students off on their own.

Overall, this site is excellent for introducing students to SQL SELECT statements and providing them feedback. It is less helpful for assessed exercises, and not useful for queries against any arbitrary database.

MYSQL QUERY BROWSER

Overview

The MySQL Query Browser [2] is a multi-platform application that allows a user to query and manipulate MySQL databases over the Internet. The application has pre-built binaries for MS Windows, Mac OS X, and several Linux distributions, as well as source code for compilation under other operating systems.

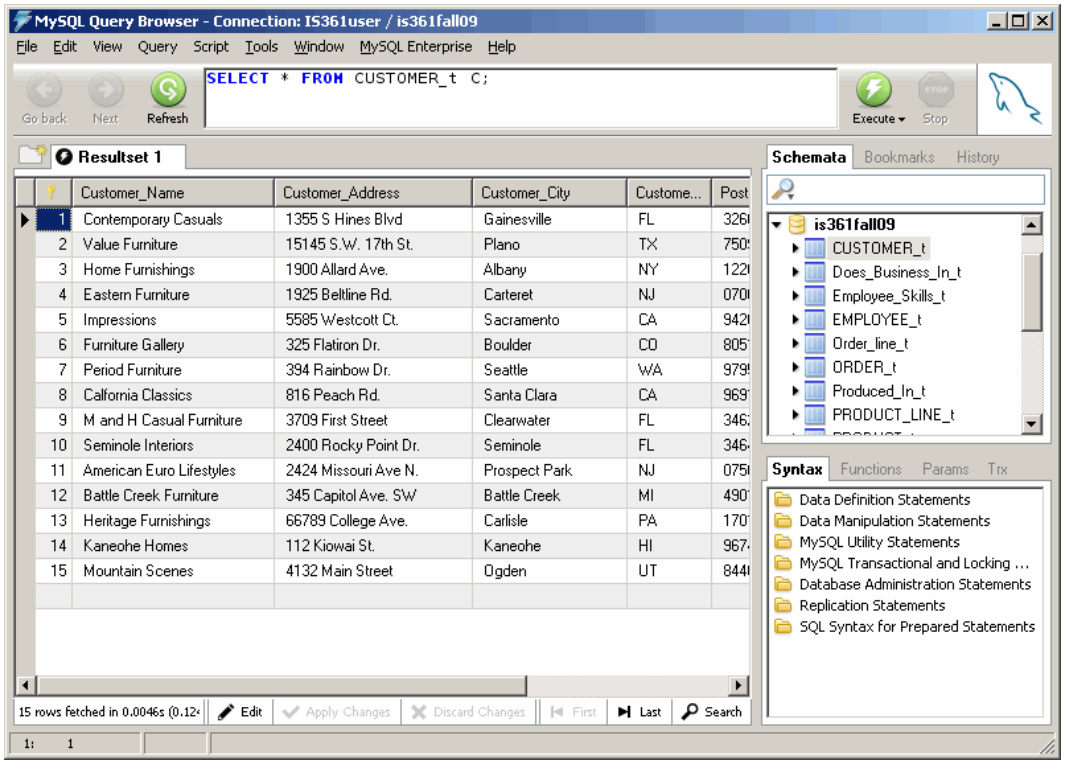


Figure 3

Strengths

The primary benefits of using the MySQL Query Browser are the graphical user interface and the total ability to query and manipulate an accessible MySQL database from anywhere on the Internet. This tool allows a user or group of users to write and test SQL and to visually see all the effects of any database manipulations. Group access is possible because the instructor or IT department controls the MySQL server. It also allows for the creation and execution of stored procedures, granting of privileges, and other database activities. Figure 3 shows an example.

Queries are saved in a history until cleared, and individual result sets can be saved to disk in a variety of formats (.html, .xls, .csv, and .xml).

Limitations

The biggest limitation is the overhead in setting up the MySQL databases and the installation of the MySQL Query Browser environment. There must be an Internet accessible server and the instructor or IT support staff must create the individual databases and accounts for each student. While not an impossible task, it does take some effort, especially initially, and must be done for every new class of students.

More problematic is the installation and configuration of the MySQL Query Browser by the individual students. The installation of pre-built binaries itself is not difficult, but many students have problems with installing software (either technically, or administratively when trying to do class work on their company's computers) Furthermore, no matter how detailed the instructions, there are always issues in getting the correct username, password, and connection port to work together for a successful login.

Overall, the MySQL Query Browser is the most flexible, complete, and usable tool of these three for learning SQL. It is also the one with the highest overhead both in terms of student installation and in terms of instructor/staff setup.

TERADATA SQL ASSISTANT/WEB EDITION

Overview

The Teradata SQL Assistant/Web Edition is a browser-based environment functionally similar to the MySQL Query Browser. The environment is accessible from any web browser with an Internet connection. Using it requires multiple registrations by the instructor, then a separate registration for each student. However, the student registrations are automatically approved. There is no additional software to install. An example session is shown in Figure 4.

Note that the Teradata SQL Assistant/Web Edition is just one resource available from the Teradata University Network [5] which also has many other free resources for learning about databases.

For those wishing to see this environment, there is a trial version that does not require registration [4]. This trial only allows read access against a single, pre-defined database but does show off the full feature set of the tool.

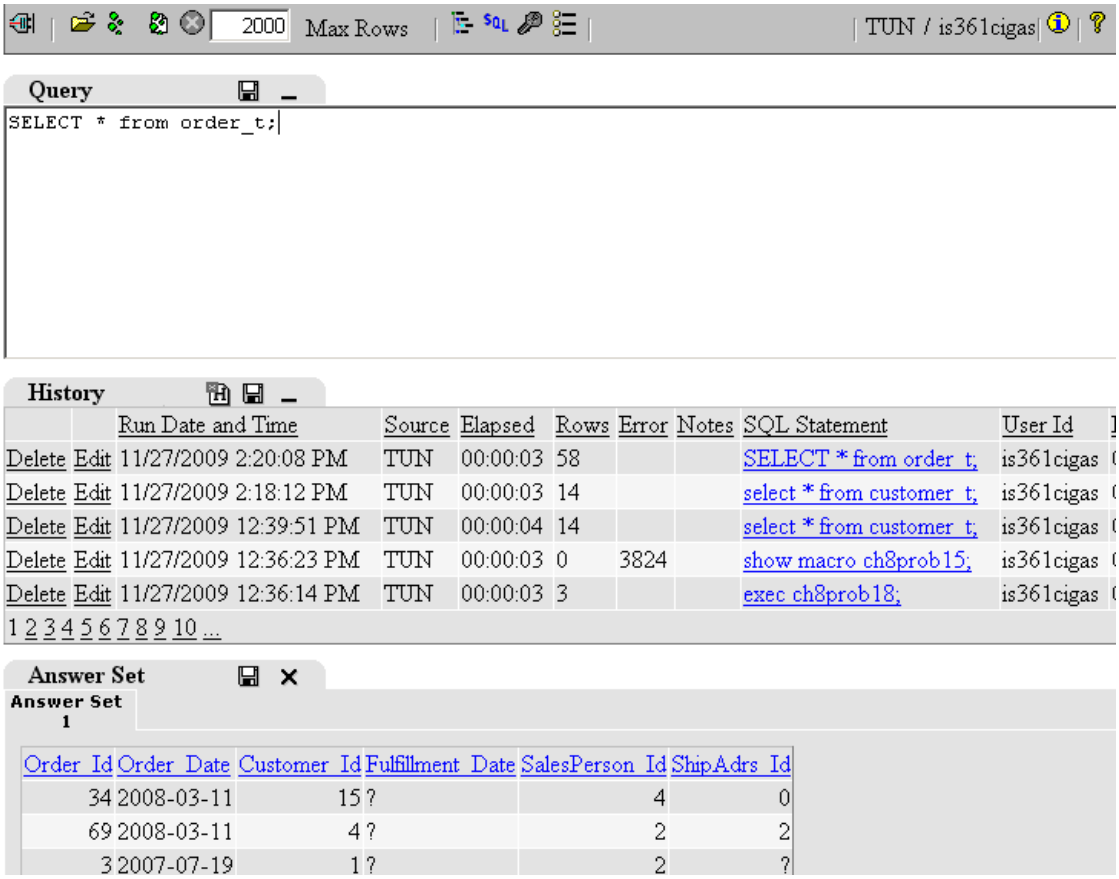


Figure 4

Strengths

The full version allows students to query a number of pre-defined databases from several popular database textbooks. The instructor has control over which databases students have access to. This read-only access is good when students are initially learning queries. For more advanced SQL, including views, stored procedures, and table manipulations, students are allowed to create their own, private sets of tables.

As with the MySQL Query Browser, students can save their individual result sets locally on their own computer in a variety of formats. Query history is retained in the online environment across login sessions. This history can also be saved locally, but this is only useful for short queries. Since student registration is tied to a specific course, this allows the system to set permissions for the instructor to have read access to all students' databases and stored procedures.

Limitations

Although this tool is slower than the locally run MySQL Query Browser, it would otherwise seem to be an ideal tool for online use, since it does not require students to install any additional software, nor fight with configuration settings. However, in practice, there are some issues that make this less than ideal for assignments. The biggest

issue is lack of support. The instructions for requesting support of the SQL Assistant/Web Edition are not correct, and requests to the main Teradata University Network site are often unacknowledged and ignored.

Having support is necessary because there are numerous issues, both large and small. These include the site being down, an instructor losing access to student work, multiple back-end servers with significantly different clocks, causing queries to be listed out of sequence, and the artificial limitation of being able to only save the first 100 characters of a query to a local file. While none of these are detrimental to learning about SQL, they are necessary for assessing student work.

CONCLUSIONS

This paper has shown three different online SQL environments, SQLzoo.net, MySQL Query Browser, and the Teradata SQL Assistant/Web Edition. It has described each, and listed the usefulness and limitations of each one in the context of online instruction. None are perfect for every situation, but all are useful in specific instances.

REFERENCES

[1] Cumming, A., SQLzoo, 2009, www.sqlzoo.net, retrieved November 27, 2009.

[2] Sun Microsystems, MySQL Query Browser, 2008, dev.mysql.com/doc/query-browser/en/index.html, retrieved November 29, 2009.

[3] Teradata Corporation, Teradata Database for Your Classes, tunweb.teradata.ws/tunstudent/, retrieved November 27, 2009.

[4] Teradata Corporation, SQL Assistant/Web Edition Trial, tunweb.teradata.ws/trial, retrieved November 27, 2009

[5] Teradata Corporation, Teradata University Network, 2009, academicprograms.teradata.com/tun/, retrieved November 27, 2009.

USING TOPIC MAP TO CREATE AN E-LEARNING ENVIRONMENT: THE TOPIC OF OSI MODEL MAP*

Marcos S. Pinto
NYC College of Technology (CUNY)
Computer Systems Dept.

ABSTRACT

Topic maps can be used to create ontology-based e-learning repositories that manage learning resources. The objective of this paper is to create a repository that enables e-learners to access and retrieve information on the definition of the layers of the OSI (Open System Interconnection) network model. The process of creating a topic map on the OSI can then be also applied to any other topic of interest which results in the gathering of all available semantic metadata on the topic and subtopics to ultimately facilitate searching and retrieving information on them. One of the key strategies for effective e-learning explorative learning is to promote student-centered learning environments, providing access to alternative resources through knowledge management processes. Topic map is such a tool providing management of learning resources.

1. INTRODUCTION

The important components of knowledge management are people, content, culture, process and technology. In a learning context, people, learners and teachers, create, share, and re-use knowledge. Content is the quantitative aspect of a specific, relevant, and authenticated knowledge that is being shared and managed. Culture must include sharing knowledge in its definition. Processes are knowledge operations of acquisition, organization, authentication, and retrieval - crucial operations for effective transfer of knowledge for both learners and teachers. Technology fosters and enhances learning experiences, and topic maps are an example of technology used for e-learning.

An e-learning experience consists of an unsupervised, self-centered, self-guided, and above all, self-documenting learning experience. One of the key strategies for effective e-learning explorative learning is to promote student-centered learning environments, providing access to alternative resources through knowledge management processes. Topic map is such a tool providing management of learning resources.

Topic map is a drawing (map) of a collection of related topics. These topics are linked to each other via drawing lines according to their relationship, whether is unary or n-ary. A topic may also be related to any number of resources by its occurrences. Topics can have various names, but also can contain references to resources outside the map (mostly URLs). This collection represents a concept, an idea, an application, an object - that is, anything that is known and the parts that compose it are also known. The map assists the learner, through navigation or query, to understand what is depicted in a compact way by revealing all its components.

The map shows the relations between topics and the relations between a topic and the resource(s). Thus topic maps are like index lists which consist of lists of topics, associations between topics (e.g., subentries), and occurrences of topics (pointed to via locators: page numbers, section numbers, etc). The fundamental constructs of a topic map are topics, associations, and occurrences, frequently mentioned as the TAO of topic maps. Furthermore, since ontologies are explicit descriptions of concepts in a specific domain and shared understanding, then it is safe to explain topic maps as a visualization of an ontology.

We are faced, at present, with the challenging task of managing the world's largest knowledge base, the world wide web (www). Topic maps simply help us to find what we are looking for. Its visible display of resources specific of certain knowledge domain make it easy to access information on the domain and discover other knowledge domain(s) related to the one at hand. Furthermore, topic map is a tool that we can rely on because it is an ISO (International Organization for Standardization) standard for representing knowledge structures[1].

The most common use for topic maps right now is to build web sites that are entirely driven by the topic map, in order to fully realize their information-finding benefits. The topic map provides the site structure, and the page content is taken partly from the topic map itself, and partly from the occurrences. This solution is perfect for all sorts of portals, catalogs, site indexes, and so on. Since a topic map can be said to represent knowledge about the things it describes, topic maps are also ideal as knowledge management tools.

This is by no means all topic maps can be used for, however. They can also be used to organize the content in content management systems (instead of the simple folder hierarchies and property-value metadata often used today), they integrate information from diverse sources (using merging), they can drive expert systems, and much more.

Topic Maps and RDF (Resource Description Framework) [5] are web technologies that can be alternatively used for knowledge management, web portal development, information search, content management and e-commerce. RDF, as its name implies, is a framework for describing and interchanging metadata. It is a language for representing information about resources on the World Wide Web, particularly intended for representing metadata about Web resources, such as the title, author, and modification

date of a Web page, However, by generalizing the concept of a "Web resource", RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web.

There are other forms of maps for e-learning such as mind maps and concept maps. These type of maps present some similarities and some differences from topic maps. Like topic maps concept maps emphasize and describe the relationships between concepts, mind maps do not; concept maps do not need to have a central topic, mind maps usually do, and topic maps require at least one; concept maps have a non-hierarchical net structure, mind maps take the form of a modified tree diagram, while topic maps are hierarchical structures. Some argue that concept maps consist of named nodes and labeled arcs. Topic maps consist of a few more objects than that, objects that deal with occurrences, scopes, roles, association membership, and so forth - more semantic objects.

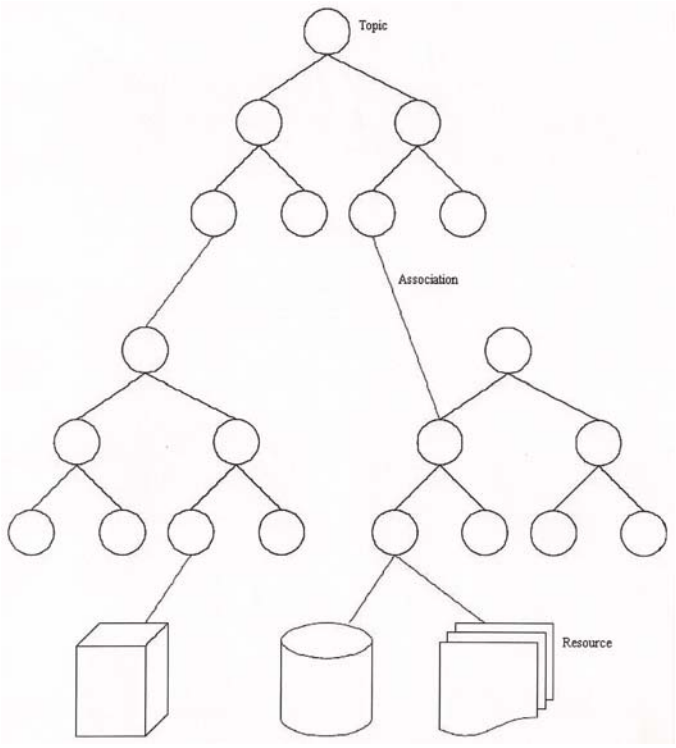


Figure 1. Topic Map and Resources

2. TOPIC MAP DEVELOPMENT

Before building a topic map, let us list the benefits of topic maps which are:

- easy visualization
- easy extension of ontology without changing the storage schema
- integrating knowledge domains: merging of topic maps
- reuse the data for other purposes
- Extensibility: more information can be incorporated at any time

Creating a topic map is easy once you classify the subject into topics and subtopics. Topics are first fed into the software according to their hierarchy and their relationships

(associations). Topics are then linked to resources (occurrences). Topic map is saved as .xtm file and it is created according to the Topic Maps Reference Model [3]. We are using the TM4L editor [2] for creating our topic map for the OSI model. Figure 1 is a generic visualization of a topic map with its topics and resources.

The map is like an index of resources related to the main topic like the index of the back of a book. The topic we chose is the layers of the OSI network model. It is a fairly known network model and its layers' definitions are the base of all the telecommunication networks around the world. The map is useful for recognizing relationships of knowledge about the OSI model and its components.

The idea is to implement the creation of topic map with a standard subject so to emphasize the efficiency of topic maps in providing a "complete as it can be" understanding of the subject depending on the willingness of the map creator and collaborators to further the breadth of the map. It should be noted that any topic map's validity is the sole responsibility of the topic map creator who is assumed to be the expert in the main subject.

3. THE DESIGN OF OSI TOPIC MAP

The standard OSI network model consists of seven layers. Table 1 shows the seven layers and their definitions according to ISO.

Layer	Name	Definition
1	Physical	Media interface, transmission method, signal strength
2	Data Link	Physical addressing, error detection, acknowledgments
3	Network	End-to-end delivery, logical addressing, routing
4	Transport	Fragmentation/sequencing of data
5	Session	Controls conversations/sessions
6	Presentation	Data formatting, i.e. ANSI, compression/encryption
7	Application	Provides services/protocols to applications

Table 1. The seven layers of the OSI network model

The topic and subtopic structure of the OSI model can be written as:

Topic: OSI

Sub-Topics: Components (Physical, Data Link, Network, Transport, Session, Presentation, Application)

Protocols (Physical Layer Protocols, Data Link Layer Protocols, Network Layer Protocols, Transport Layer Protocols, Session Layer Protocols, Presentation Layer Protocols, Application Layer Protocols)

The next thing is to create the associations (relationships) between the topics. We make use of two associations: the whole-part and the placed-atop. The former is of the type includes/part-of-it, and the latter is of the type placed-atop-of/placed-below-of. The whole-part is a typical relationship of topics that are related to each other hierarchically. The seven layers of the OSI model communicates with the adjacent layers, therefore the

association placed-atop-of would recognize this situation. For example, it can be formulated that the topic Application Layer is part of the OSI model and it is place on top of the Presentation Layer which is also part of the OSI model. This relationship can be seen in the topic map of the OSI model shown below.

We will implement the resources for better understanding the OSI model using web-based tools such as URLs, blogs, and wikis.

4. TM4L EDITOR

We use the TM4L Editor which is a free software that allows us to create topic maps. It consists of a TM editor and a TM viewer. The TM4L Editor is an ontology editor allowing the user to build ontology-driven learning repositories using Topic Maps. It provides ontology and metadata engineering capabilities coupled with basic document management facilities.

The TM4L Editor benefits from the Topic Maps' fundamental feature to support easy and effective merge of existing information resources while maintaining their meaningful structure. This allows for flexibility and expediency in re-using and extending existing repositories.

The learning content created by the editor is compliant with the XML Topic Maps (XTM) standard and thus interchangeable and interoperable with any standard TM tools.

A screenshot of the TM4L Editor interface is shown on Fig. 2.

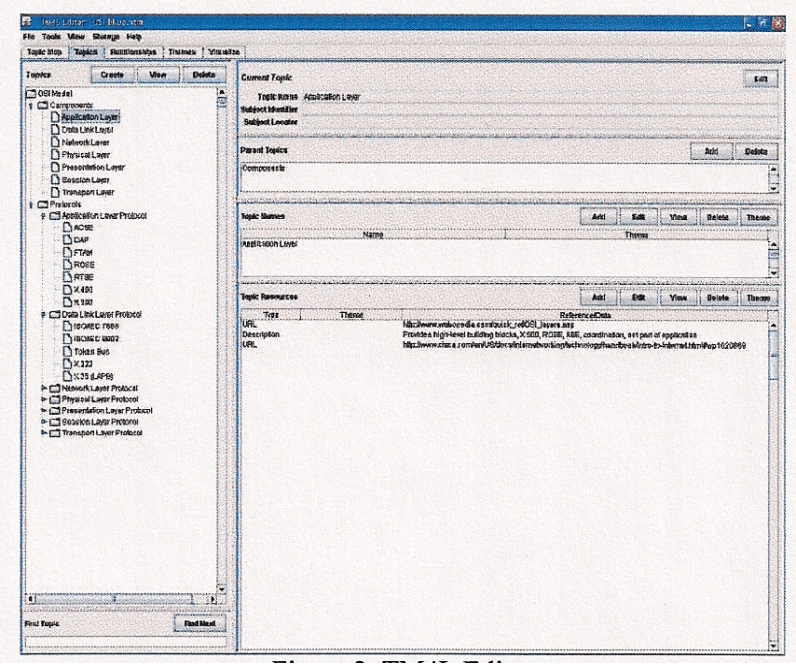


Figure 2. TM4L Editor

The TM4L Editor is implemented as a client-server application developed in Java and using the TM4J Topic Map Engine, which is an open source providing comprehensive API that allows creating and modifying topic map structures.

The associations of whole-part (instance-of) and placed-atop are defined for each of the seven layers of the OSI model. Figure 3 shows the relationships between topics.

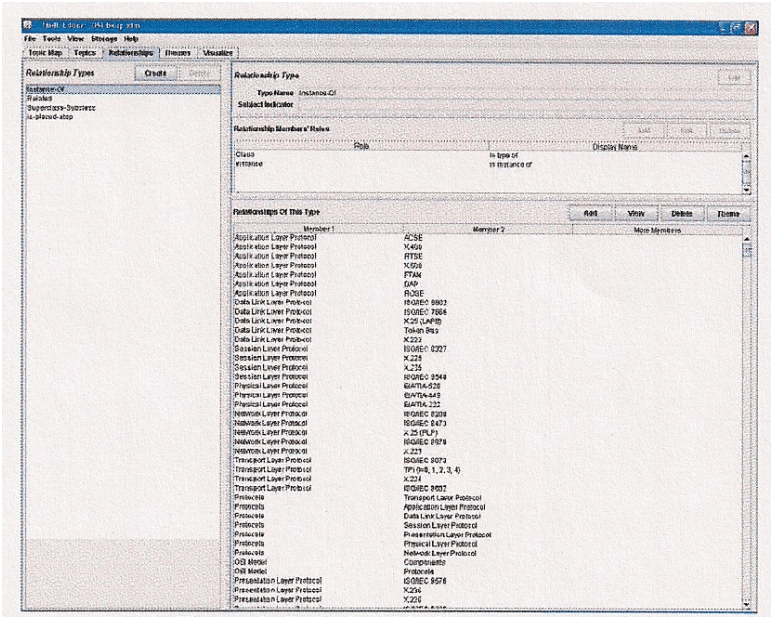


Figure 3. Relationship between topics

5. CONCLUSION

Topic maps are a form of semantic where the knowledge base is driven by visual features. These characteristics support our idea of providing an interpretive, semantic layer on top of document collections that classify these documents according to scope, context and constraints. Topic map can be recommended as a support tool in educational planning in one more way: it can bring to light the concepts that pass through several distinct topics. The visual form of knowledge representation must also be mentioned: without this, teachers would need much more time and effort to show their own schemes to students and fellow teachers. It provides better personalized courseware presentations using visual semantics. It can help distributed courseware development for the reuse and the exchange of learning materials. Finally, it is possible to set collaborative visual topic maps authoring among instructors and/or learners. Work is currently being undertaken to specify multi-dimensional metadata under topic maps model and to collect related data from experts and pupils. The next step will be for experts and non-experts such pupils to enrich visual topic maps with sets of vast amount of single documents or learning objects of the same context to make them more easily accessible.

6. REFERENCE

[1]. ISO (2005) "ISO/IEC CD 13250-5 Topic Maps - Part 5: Reference Model". Online at http://www.isotopicmaps.org/TMRM/TMRM-5.0/TMRM-5.0.pdf

[2] TM4L: Topic Maps 4 e-learning. http://compsci.wssu.edu/iis/nsdl

[3] Dicheva, D. and Dichev C. (2007): Authors support in the TM4L environment. *Information Technologies and Knowledge* 1(3):215-218.

[4] D. C., D. Dicheva, and L. Aroyo(2004). Using topic maps for web-based education. *Int. J. of Advanced Technology for Learning*, volume 1(1), pages 1-7, 2004.

[5] Berners-Lee, T. Hendler, J. & Lassila, O., The Semantic Web. *Scientific American*, 17 May 2001.

STUDENT CLASSROOM SOFTWARE DEVELOPMENT PROJECTS: A PRACTITIONERS PERSPECTIVE*

PANEL DISCUSSION

Edward Mirielli
Westminster College, Fulton, MO
ed.mirielli@westminster-mo.edu

Kian L. Pokorny
McKendree University, Lebanon, IL
klpokorny@mckendree.edu

James Buchan
College of the Ozarks, Point Lookout, MO
buchan@cofo.edu

The development of software involves the strategic integration of processes. These sorts of projects are integrative at all stages of the system development life cycle and concern issues of available technology services, infrastructure, components, as well as people. Uncertainty in the outcome and scope of the project can be very high during the inception stage. Being systematic at the outset and throughout a software development project provides for a better understanding of the scope of the problem to be addressed, a more applicable vision for a solution, and an empirical estimate of the overall project feasibility and success. Due to these and other issues encountered when developing software, adopting a software engineering approach has become an increasingly important topic of debate in the computer science and software engineering literature [3,4,5,6]. This debate has a direct impact on the application of model curriculum and ideological principles advocated as "best-practice" in CS and SE education.

As with most topics in computer science, the scope of "model curriculum" content coverage is growing faster than most departments can adapt[1]. This is particularly true for small academic departments that cannot continually add and staff new courses. Beyond the traditional aspects covered in software development and programming courses, in today's post-modern world, it is vital the students receive a more integrated learning experience which emphasize aspects of oral and written communication, team work, professional practices, and ethics, to name a few. Each learning experience, especially at the upper-level, must be directed at total student development and be instrumentally designed to pedagogically address Bloom's higher level learning constructs

of synthesis and evaluation [2] and provide a proving ground for students to practice and demonstrate their skills, ability, and creativity.

Academia is a goal-driven institution. Schools have mission statements. Departments and programs have learning goals. Individual courses have learning objectives. Too often, our students become focused on individual courses as goals in themselves instead of the overall learning objectives and the relevance of a course to their intended program of study. This panel will discuss our overall curriculum design, individual course objectives and activities, and the integrated project experiences we use to address the common themes inherent in software engineering and development projects. We conclude with a discussion of what is required from the instructors and students including implications for practice in today's CS classroom.

REFERENCES

[1] ACM/IEEE. *Computer Science Curriculum 2008: An Interim Revision of CS 2001*. November, 2008.

[2] Bloom, Benjamin S. & David R. Krathwohl. (1956). *Taxonomy of educational objectives: The classification of educational goals, by a committee of college and university examiners. Handbook 1: Cognitive domain*. New York , Longmans.

[3] Boehm, Barry. *A View of 20th and 21st Century Software Engineering*. ICSE'06, May 20–28, 2006, Shanghai, China. 2006 ACM 1-59593-085-X/06/000

[4] IEEE Computer: *Rethinking Formal Methods*. September 2009.

[5] Jianguo, LIU. *Combination of Research and Teaching in Software Engineering Education*. 2009 WASE International Conference on Information Engineering. 978-0-7695-3679-8/09 2009 IEEE DOI 10.1109/ICIE.2009.53

[6] Sjøberg, Dag I. K., Tore Dybå & Magne Jørgensen. *The Future of Empirical Methods in Software Engineering Research*. Future of Software Engineering (FOSE'07) 0-7695-2829-5/07, 2007

TEACHING AND INTEGRATING SOCIAL AND PROFESSIONAL ISSUES INTO A SMALL COLLEGE COMPUTER SCIENCE CURRICULUM*

TUTORIAL PRESENTATION

Carol Spradling
Computer Science/Information Systems
Northwest Missouri State University
Maryville, MO 64468
660-562-1588
c_sprad@nwmissouri.edu

Brian Hare
School of Computing and Engineering
University of Missouri - Kansas City
Kansas City MO 64110
816-235-2362
hareb@umkc.edu

ACM and IEEE support the inclusion of social and professional issues in the computer science curriculum in the 2001 Computing Curricula Computer Science volume and the recent C2008 Computing Curricula Computer Science volume. The IEEE-CS/ACM Joint Task Force on Computing Curricula note:

> Undergraduates also need to understand the basic cultural, social, legal, and ethical issues inherent in the discipline of computing. They should understand where the discipline has been, where it is, and where it is heading. They should also understand their individual roles in this process, as well as appreciate the philosophical questions, technical problems, and aesthetic values that play an important part in the development of the discipline. ([4], p. 152)

The 2008 Computer Science Curriculum Interim volume [1] identifies knowledge units regarding seven required coverage topics (history of computing, social context of computing, methods and tools of analysis, professional and ethical responsibilities, risks and liabilities of computer-based systems, intellectual property, privacy and civil liberties), and four remaining topics considered appropriate for elective coverage (security operations, computer crime, economic issues in computing, philosophical frameworks).

Spradling, Soh and Ansorge [6] conducted a study of 251 computer science programs in the United States and determined that 88% (220 programs) address social

and professional issues in their courses. While this finding is encouraging, the study further uncovered that of the 31 programs not covering social and professional issues, 77% (24 programs) have enrollments under 100 computer science majors. Of the 24 programs not covering social and professional issues, the study determined that the major reason that these programs do not teach social and professional issues is because computer science faculty have not been trained [5]. Engaging students on these issues does present some challenges [2], but does not require extensive specialized expertise. Resources are readily available to assist in engaging students [3].

This workshop will address social and professional topics, how to integrate these topics into the curriculum or in a standalone courses, and explore appropriate resources available to faculty and discuss how to integrate these resources. Both of the presenters have experience developing coursework around professional ethics and responsibility, and integrating these subjects into the larger curriculum.

REFERENCES

[1] ACM/IEEE-CS Joint Curriculum Task Force. (2008, December). Computer Science Curriculum 2008: An Interim Revision of CS 2001. Retrieved Novebmer, 2009 from http://www.acm.org//education/curricula/ComputerScience2008.pdf

[2] Hare, B. (2008). Ethics education in computer science: engaging future technologists on policy issues. Presentation at Oxford Roundtable: "Regulating the Internet: Balancing the Interests," March 19, 2008, Pembroke College, Oxford University, Oxford, England.

[3] Hare, B (2009). Implementing a writing-intensive C.S./I.T. ethics course. Journal of Computing Sciences in Colleges, (24)1, 76-82.

[4] IEEE-CS/ACM Joint Task Force on Computing Curricula. (2002). Computing Curricula 2001 Computer Science. (Final Report IEEE-CS/ACM Joint Task Force on Computing Curricula). Los Alamitos, CA: IEEE Computer Society.

[5] Spradling, C., Soh, L.-K. & Ansorge, C. (2008). Ethics training and decision-making: do computer science programs need help? ACM SIGCSE Bulletin, 35(1), 153-157.

[6] Spradling, C., Soh, L. & Ansorge, C. (2009, September). A comprehensive survey on the status of social and professional issues in United States undergraduate computer science programs and recommendations. Computer Science Education Journal, 19(3), 137-153.

CLUSTER COMPUTING ON A SMALL DEPARTMENT BUDGET*

David Bainum
Washburn University, Topeka KS
(785) 670-1261
david.bainum@washburn.edu

Bruce Mechtly
Washburn University, Topeka KS
(785) 670-1160
bruce.mechtly@washburn.edu

ABSTRACT

This paper discusses our experience in an advanced computer science course in which students built modern computing clusters from scratch. The least expensive cluster used an existing computer lab for the compute nodes. The students also built a 10-node cluster using bare-bones computer kits for a total cost of about $2800. For software we used Perceus which is included in the CAOS NSA Linux distribution. Students were required to perform a number of administrative tasks on the system, write and test simple parallel programs (using OpenMPI), and run benchmark programs to compare performance with other parallel systems.

INTRODUCTION

Cluster computing is a somewhat advanced field for undergraduates, but promises to be an increasingly important one in the future. Stories of large computing facilities abandoning mainframes in favor of computing clusters are now commonplace. While state-of-the-art clusters are typically built using the very best computing hardware, there are many instances where surprisingly powerful systems are built using ordinary desktop systems. These clusters are sometimes referred to as Commodity, Off-The-Shelf (COTS) systems [1]. The obvious advantage of a COTS cluster is that it can be built cheaply; making it perfectly suited for an academic environment.

In what follows we will describe our experience building, configuring and testing COTS clusters. The CAOS NSA Linux operating system was installed on the master

node and the CAOS NSA Node operating system downloaded into the memory of the compute nodes at boot time. The Perceus Cluster Management Software is included in the CAOS NSA system installed on the master node.

Students successfully built Perceus clusters in two different environments. First, existing computers in a lab were used as the compute nodes. Only one additional system was required to install the master node OS. The computers in the lab PXE booted and downloaded their operating system image from the master. The default downloaded OS image does not mount the local hard disks, so the systems in the computer lab are not disturbed except for BIOS changes to start with PXE boot.

The second cluster the students constructed required the purchase of 10 bare-bones kits. One kit served as the master node and the other 9 served as compute nodes. These computers were dedicated to use as a cluster. In this environment students could do more advanced operations such as configuring separate data and control networks and mounting the local hard drive for scratch space.

HARDWARE

We purchased 10 identical systems. They were sold as "bare-bones kits". They included the case, power supply, motherboard, CPU (dual core Pentium), 1 gigabyte of RAM and a 160 gigabyte hard disk. A CPU fan and a case fan were purchased separately. The systems were purchased from Tiger Direct [2] for about $220 each. Another good source is New Egg [3]. These systems were sold without any operating system, which is perfect for building a Linux cluster.

If you are not using an existing computer lab you will also need to get at least one ethernet switch (~$200 for a layer 2 switch), an extra ethernet card for the master node (if you want to access your cluster remotely) and at least 10 ethernet cables. You can get better performance by buying two ethernet switches and extra ethernet cards and cables for the nodes. Later in this paper we will discuss how to set up separate data and control networks to help with performance.

OPERATING SYSTEMS

For our cluster operating system we chose CAOS NSA Linux [4] which includes Perceus [5] as the cluster management software. CAOS NSA (Node, Server, Appliance) is a Linux distribution developed for servers, compute nodes and network appliances. CAOS NSA was installed on the master node of the cluster. As part of the CAOS installation, Perceus was selected for installation and configuration for cluster management. CAOS uses dnsmasq for DHCP and TFTP servers which PXE boot the compute nodes. Immediately after booting, the compute nodes are provisioned with a Virtual Node File System (VNFS) by Perceus running on the master node. The CAOS NSA Node system was used as the VNFS in this cluster. The VNFS is an operating system image which is downloaded into the memory of the compute nodes from the master node via a provisioning daemon running on the master. The obvious advantage of using Perceus is that there is no need to individually install an operating system on the compute nodes.

Perceus provides a number of tools to set up and administer the cluster. Particularly important at the outset is to establish the identity of each compute node and set up communication with each node using ssh. Also, it is typical in clusters to mount the home directories, located on the master hard drive, on each compute node via NFS. When users are created on the master, Perseus creates a set of public and private keys to facilitate logging into the computer node via ssh, should one want to login to the compute nodes. Program development using C and C++ and OpenMPI can be done in the user's directory on the master so there is typically no need to log into the compute nodes.

CAOS AND PERCEUS INSTALLATION

By far the easiest way to install Perceus is to download the iso image file for CAOS NSA Linux [4], which includes Perceus, and burn that iso image to a CD.

After booting the master node using the CD, the CAOS NSA install script will guide you through the installation, which is fairly straightforward. You will need to provide identity information for the master node. When it gets to "Select roles for this system" you should select "File-Server" and "Clustering". When it asked if Perceus should be installed select "Yes".

When you get to "Perceus Configuration" you have the ability to choose the number of the first node the total node count and a naming scheme for the nodes (we used the default n####). The compute nodes are usually on their own private network typically 192.168.1.X or 10.1.1.X. We chose 10.254.0.0/22 as the subnet for the cluster.

Perceus configuration will also ask you to select a VNFS image for use on the compute nodes. The installation will suggest a VNFS image which will be automatically downloaded from the Internet if it is selected. In our installation, the VNFS image downloaded was caos-nsa-node-1.0.25.X86_64.vnfs. The downloaded VNFS was stored in the /var/lib/perceus/vnfs directory on the master node. This VNFS image will be downloaded into the compute nodes immediately after booting via PXE. In preparation for PXE booting the cluster compute nodes, the master is configured as a TFTP and DHCP server using dnsmasq. The files to support PXE booting of the compute nodes are stored in /var/lib/perceus/tftp. Perceus and dnsmasq configuration files are located in /etc/perceus.

SETTING UP THE COMPUTE NODES

The compute nodes and master node were connected to a 24-port gigabit ethernet switch which was the high-speed interconnect (HSI) [6] of the cluster. The BIOS in the compute nodes was set to PXE boot from the master node when the compute nodes are powered on.

Assuming a successful install of CAOS NSA and Perceus on the master node, the compute nodes can now be PXE booted and obtain an IP address from the DHCP server on the master. Immediately after booting, the compute nodes are provisioned when the master downloads, using NFS, the VNFS image into the memory of the compute nodes. The master monitors the provisioning and records the state of the compute nodes as the state changes from an initial "init" state while being provisioned to a final "ready" state.

REBUILDING THE IMAGE FILE

The VNFS image, which is the operating system downloaded into the compute nodes when they are booted, can be customized by mounting it on the master using the "perceus vnfs mount" command. Once mounted, the VNFS can be accessed as a file system and changes easily made. Also, packages can be added to the VNFS by using the CAOS SMART package management tool. After the desired changes have been made, the "perceus vnfs umount" command will rebuild the VNFS image. Rebooting the compute nodes will load the modified VNFS image or the command "perceus vnfs livesync" can be used to load the modified VNFS image into the compute nodes without rebooting them. For example, the java RTE was installed into the VNFS so it would be available on each compute node. Also, for test purposes the VNFS was modified to allow the root user to log in locally on each compute node.

USING PDSH AND OPENMPI

To execute code on the compute nodes, the parallel distributed shell, pdsh, can be used to manually copy the code to each compute node or OpenMPI can be used to develop MPI-based applications which are distributed to the compute nodes using MPI library routines. As an example of the use of pdsh, a java application which calculated members of the Mandelbrot set was copied to each compute node so that multiple Mandelbrot images could be computed in parallel to make a Mandelbrot "movie". The first MPI program [7] students run is a "hello world" program which has each compute node report in the compute node identity and the rank of the process executing on the node. Other MPI programs to calculate integrals and perform other parallel computations were written by the students who finished the semester with a MPI project of their choosing. Students also ported benchmark programs to the cluster to test the performance of the cluster and its high-speed interconnect.

SEPARATE DATA AND CONTROL NETWORKS

In its initial configuration, the cluster high-speed interconnect (HSI) was one gigabit ethernet network for control and communication and data distribution. After that configuration was working and throughly tested, the HSI was split into two gigabit networks, one for control and communication and one for data distribution using NFS. All compute nodes had two ethernet interfaces, one on the motherboard and one as a PCI adapter. The master node had three ethernet interfaces; one for the command and control network, one for the data network and one for access to the public Internet. The control and communication subnet IP addresses were assigned using DHCP. The ipaddr Perceus module was edited to enter the data for the subnet IP addresses.

For example, to create separate control and data subnets of 10.254.0.X/22 and 10.254.4.X/22 the ipaddr file looked like this:

```
* eth0:[default]/[default] eth1:10.254.4.1/255.255.252.0/10.254.4.1
n0000 eth0:[default]/[default] eth1:10.254.5.0/255.255.252.0/10.254.4.1
 . . .
```

MOUNTING THE LOCAL HARD DISK

Since each compute node had a 160 gigabyte hard disk, the VNFS was modified to mount that disk locally on each compute node when the node was booted. This allowed for local "scratch" space and for storing data sets which were distributed prior to executing code utilizing the data sets. To do this the students edited the file /etc/fstab in the VNFS image.

CLUSTER OPERATION AND ADMINISTRATION

The Perceus Cluster Management System was installed as part of the CAOS NSA installation on the master node. Students used Perceus commands to perform routine systems administration tasks on the cluster. The following is a summary of Perceus commands which the students used to build and administer the cluster [8]:

perceus node add	Import a new node into the node database
perceus node delete	Delete a node or set of nodes
perceus node list	List the configured nodes
perceus node show	Dump the node configuration
perceus node status	Current status of node provisioning
perceus node summary	General configuration summary
perceus vnfs import	Import a new VNFS capsule into Perceus
perceus vnfs list	List configured VNFS capsules
perceus vnfs livesync	Update the file system of a running node
perceus vnfs mount	Open the VNFS image to modification
perceus vnfs umount	Close the VNFS image and save any changes
perceus module activate	Activate a module to run in a provisionary state
perceus module list	List the installed modules (doesn't mean active!)
perceus module summary	Show list of modules and activated states
perceus info system	Print system information (useful for troubleshooting)

MEASURING PERFORMANCE

The IOZone [9] and Skampi [10] benchmark programs were used to measure cluster performance. IOZone was run across a wide range of record sizes and file sizes to test data communications throughout the cluster. Skampi included "pingpong" functions which measured point-to-point communications in the cluster.

An MPI version of the N-queens program from Kise [11] was used to measure the computational performance of the cluster and compare the cluster with other clusters. For this purpose, a lab with 20 PCs was "clusterized" using CAOS NSA and Perceus to test the effect of having more compute nodes in the cluster. Each node had an AMD Athlon(tm) 64 X2 Dual Core Processor with a 3 GHz clock speed and 2 gigabytes of memory. This cluster used a 100 Mbps HSI and the command and control and data networks were combined on one switch. This 20-node cluster is identified as the "Mo16" cluster while the 10-node cluster discussed previously is the "St314" cluster. The following is a comparison between these two clusters using the MPI-based N-queens program:

Cluster	Number of Queens	Time to Find All Solutions (seconds)
St314	19	551.3
Mo16	19	279.5
St314	20	4066.3
Mo16	20	1888.4

The execution time for this application scales with the number of compute nodes.

CONCLUSIONS

Building a cluster system is very easy using the CAOS NSA Linux distribution and the Perceus Cluster Management System. The developers of Perceus have used their Warewulf experience to develop a powerful yet easy to install and operate cluster management system. Inexpensive computers constructed from "bare-bones" kits can be used for the master node and compute nodes. It is possible to include a CAOS NSA server in an existing computer lab to "clusterize" the computers in the lab. Thus, anyone with a small department budget and/or a computer lab can have a cluster by using the "magic" of Perceus.

REFERENCES

1. Sloan, Joseph D., *High Performance Linux Clusters*, O'Reilly Inc., 2005, p. 8.
2. Tiger Direct Inc. 2009, www.tigerdirect.com.
3. Newgg Inc. 2009, www.newegg.com.
4. The CAOS Project, 2009, www.caoslinux.org.
5. The Perceus Project, 2009, www.perceus.org.
6. Lucke, Robert, *Building Clustered Linux Systems*, Prentice Hall PTR, 2005, p. 22.
7. Sloan, Joseph D., *High Performance Linux Clusters*, O'Reilly Inc., 2005, p. 228.
8. The Perceus Users Guide, www.perceus.org/docs/perceus-userguide-1.5.0.pdf.
9. IOzone Filesystem Benchmark, 2009, www.iozone.org.
10. SkaMPI 5 Benchmark Software, 2009, liinwww.ira.uka.de/~skampi.
11. Kenji Kise, et. al., *Solving the 24-queens Problem using MPI on a PC cluster*, Technical Report UEC-IS-2004-6, Version 2004-06-14.

A ONE-CREDIT ARTIFICIAL INTELLIGENCE COURSE FOR A GENERAL AUDIENCE*

Michael Black
American University
mblack@american.edu

ABSTRACT

In order to expose more students to computer science, we developed "Artificial Intelligence", a one-credit course intended for non-computer science majors. The course seeks to give students a taste of various technical topics in AI, ranging from expert systems to neural networks, without requiring any programming or mathematics background. The course uses a mixture of hands-on exercises and software demonstrations to illustrate the principles behind AI. Despite the technical subject matter, students from a variety of majors reported that they found the course both accessible and interesting.

1. INTRODUCTION

With enrollments declining, luring students to computer science is a serious challenge. Our computer science program relies heavily on recruiting internally, encouraging current students who are undecided about their major to consider computer science. One of our key problems is that most students are unfamiliar with computer science and have misconceptions about the subject. To remedy this, we have started to offer one-credit "fun" courses intended for nonmajors. These courses have no prerequisites and require no prior experience with programming or mathematics. However, they are nevertheless intended to give students a comprehensive introduction to an area of computer science.

This paper describes "Artificial Intelligence", the first course we offered. Artificial intelligence lends itself naturally to a general survey course. It is relevent: all students have unknowingly come into contact with AI software on the internet, in games, or elsewhere. It has a dark and controversial side, as any student who has seen *The Matrix*

or *Blade Runner* can attest. AI can also be humorous (think *Dr. Eliza*), and lends itself well to hands-on demonstrations.

Courses abound both at our university and elsewhere that discuss AI's effect on society, usually in the context of science fiction. However, in designing this course, I opted for a different approach. Rather than focus on the social impact of AI, I instead looked at the technical aspects of AI. The resulting course explored various tricks and approaches to mimicking AI, from neural networks to expert systems to exhaustive searching. "How does it work?" was the underlying question.

There are several reasons for making a technical survey course. First, students can already find discussions of AI and society in existing courses. Second, students are unlikely to have their misconceptions of computer science dispelled unless they are presented with real computer science. Third, a course designed to entice students to study computer science should show students examples what they might study.

Creating an AI course with broad appeal and a technical subject matter turned out to be a balancing act. On one hand, the course had to appeal to the nonscience student. On the other hand, much of AI is mathematically dense and deals with abstractions. I approached this by creating a course that avoids theory as much as possible and focuses instead on hands-on demonstrations and simple explanations. I wrote a series of software demonstrations explicitly for the course.

Survey results from students at the end of the course were universally positive. Students enthusiastically took part in discussions and exercises, in some cases bringing their own material to class. A second offering of the course, advertised largely by word-of-mouth, filled rapidly. Several students subsequently pursued independent study projects based on the course material.

In this paper I will discuss how I structured the course and approached the material, lesson-by-lesson. In the end I will discuss the results of student feedback.

2. THE COURSE AND ITS STUDENTS

An unusual structure was chosen for this course. Rather than running one semester-long section, we opted for two sections, offered back to back. The first section started in late January and concluded in early March. The second section began in March and concluded at the end of April. We chose this structure for several reasons. It allowed us to gauge student interest by seeing how many students enroll for the second section, which was advertised by word-of-mouth. The second section also gained us students who dropped another course early in the semester and needed to pick up a credit-hour. Because of the short duration of the course, weekly class periods lasted 105 minutes, allowing a new topic to be covered in entirety each week.

14 students enrolled in the first section and 10 in the second section. A variety of majors were represented, including Math, Theater, Psychology, Journalism, and Economics. 5 of our own computer science students also enrolled. Students were drawn to the class for different reasons, including flyers, their advisor's recommendation, and the urging of their friends.

The course consisted of one lesson per week for seven weeks. In each class period I covered a separate topic. I generally followed the same structure for each concept:

- A teaser software demonstration. Some were internet demonstrations and and some were written for the course.
- An exercise. Students were given a hands-on activity that illustrated the concepts to be discussed that day.
- An interactive discussion, where students learned how the technology worked.
- One or more illustrative software demonstrations.
- Examples of real applications, sometimes accompanied by additional demonstrations.

Because a grade had to be assigned, the students were required to write a brief group paper about an application of AI technology. Students were asked to find an example, discuss how the AI works, and constrast it with approaches used before the technology existed. Some of the topics students investigated included fuzzy logic, post office zip code scanners, Pleo, and facial recognition.

3. LESSONS

Five topics were covered in this course: searching, expert systems, natural language, neural networks, and genetic algorithms. Two additional class periods dealt with defining AI and the future of AI. A synopsis of each lesson is detailed below.

3.1. Defining AI

My objective for the first lesson was to show students the difficulty of defining artificial intelligence or its goals. We covered three major topics.

- "How do I know my neighbor isn't a robot?" Students interviewed neighbors and listed on an index card what behaviors demonstrated that their neighbor is human.
- The Turing Test
- A short narrative on the history of AI, from mechanical automata to the present.

Students were assigned to read *Computing Machinery and Intelligence* and discuss it briefly at the beginning of the next lesson.

3.2. Searching

The second lesson introduced students to the concepts of backtracking, search trees, and heuristics. The lesson started with mazes and puzzles, and proceeded to competitive games like Chess. Figure 1 shows screenshots of the demonstrations used.

- Mazes. Students were asked to solve a maze on paper and note down their thought process. Two maze solving programs were demonstrated, one only using an exhaustive search and the other incorporating a heuristic.
- 8 puzzle. Students were given 8 puzzles to solve by hand, and then tried several 8 puzzle solving programs that used different searching techniques. Students learned the difference between breadth and depth first search.

- Tic Tac Toe. Students played each other at Tic Tac Toe and noted how Tic Tac Toe is effectively a search problem.
- Money game illustrating minimax. A volunteer was shown three envelopes, each containing pennies, nickels, and dimes. The volunteer was told that she would receive one coin from whichever envelope she chose, but would not get to choose the coin.
- Chess. Students watched two AIs play each other, one searching to depth 2, the other to depth 3.
- Pruning. The money game was again with a 5 second time limit, teaching the volunteer to reject envelopes with pennies.

Figure 1: Search Demonstrations

3.3. Expert Systems

The third lesson introduced students to decision trees, predicate logic, and forward/backwards chaining.

- Decision trees. Students played several rounds of the "Guess the animal game" and discussed how it works. Students then played the "Guess the Dictator / Sitcom Character" where the weaknesses of decision trees are very apparent.
- Rules chaining. Students learned about how forwards and backwards chaining, and tried a simple medical diagnostic software using forward chaining.

3.4. Natural Language

In this lesson I focused on three applications: machine translation, composition, and conversation. For each application, we discussed a rules based approach and a statistical approach.

- Dr. Eliza. Students played the psychiatrist game and then looked through the rules file to discover how it works.
- Translation. Students learned about parsing and tried out Stanford's online parsing utility [] to see how it handles ambiguous sentences. We then used the Systran translator to translate Spanish news articles to English.
- Scigen. Students tried the Scigen utility to generate phony research papers.

- Markov models. Students tried out a Markov model-based sentence generator written for the class. They produced Keats-style poems, Alice in Wonderland style prose, and State of the Union addresses in the style of George Bush.
- Google translator. Students used the statistical-based Google translator and compared it with Systran.
- Jabberwacky. The class conversed with the online Jabberwacky chatterbot.

3.5. Neural Networks

Students were introduced to the concept of making computer models of brain cells to solve problems. Figure 2 shows screenshots of the neural network software written for this course.

- Perceptron. Students trained a perceptron program written for the class to learn the max function and observed how the weights responded. Students then tried training it on the XOR function.
- Multilevel network. Students taught XOR to a second program modeling a multilevel neural network.
- Tic Tac Toe. A Tic Tac Toe game that uses a neural network to mimic the human player's strategy. Students tried teaching the game to play Tic Tac Toe, and saw how it learned to mimic wrong moves as well as right moves.
- Character recognition. Students tried a Hopfield network-based program written for the course. The program asked students to draw a number and tried to guess the digit.

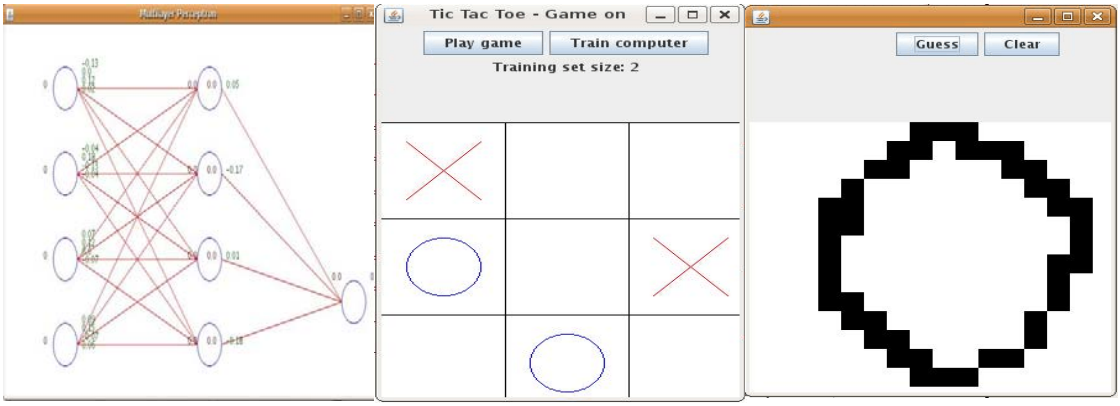


Figure 2. Neural Network Demonstrations

3.6. Genetic Algorithms

The last technical topic covered was genetic algorithms. Students discussed the principles of evolution and brainstormed how they could be modeled by a computer.

- Knapsack solver. I developed a genetic algorithm program for the course that shows the population on a GUI. Students tried solving an example problem with index cards, then watched the computer solve it.
- SRAT solver. Students tried solving James Propp's *Self Referential Aptitude* Test [] by hand, and then watched a genetic algorithm estimate a solution.

3.7. Future of AI

In the final lesson of the course I showed a series of movie clips showing various science fiction authors' visions of robots and the future of AI. Clips were chosen from *2001*, *Blade Runner*, *AI*, *the Matrix*, *the Day the Earth Stood Still*, and various *Twilight Zone* episodes.

4. SURVEY

In addition to the regular evaluation, a survey was given to the class at the end of the last lesson. Student responses to the survey questions are shown in Table 1.

Did you find this course interesting?	All students answered "yes"
Did you feel it was too technical or not technical enough?	1 answered "Too technical", 2 answered "not technical enough", 11 answered "about right"
What topics would you have liked to have seen?	"Emotional replication" and "robotics" were given.
In this class I tried to mix demonstrations, explanations, and exercises. Of those three, which would you have liked have seen more of?	7 answered "demonstrations", 6 answered "exercises"
Is there anything else you want to comment on that could be improved?	1 answered "more movies", the rest answered "nothing" or left it blank

Table 1: Qualitative feedback (first section only)

In addition, students were asked, for each topic, whether they enjoyed it, felt neutral about it, or disliked it. As can be seen in Table 2, there was no consensus, with different students enjoying very different topics.

Topic	# enjoyed	# neutral	# disliked
Defining AI	10	4	
Search problems	14		
Expert systems	11	1	2
Neural networks	13	1	
Genetic algorithms	10	4	
Movies about AI	9	3	2
Philisophical discussions			

Table 2: Feedback by topic

REFERENCES AND RESOURCES

[1] Bowles, "An Idiot's Guide to Neural Networks", http://richardbowles.tripod.com/neural/neural.htm

[2] Guzdial and Forte, "Design process for a nonmajors computing course", SIGCSE Technical Symposium, 2005.

[3] Propp, "Self Referential Aptitude Test", http://faculty.uml.edu/jpropp/srat.html

[4] Russell and Norvig, "Artificial Intelligence, A Modern Approach", 1995.

[5] Turing, "Computing Machinery and Intelligence", Mind, 1950.

[6] "Guess The Dictator / SitCom Character", http://www.smalltime.com/Dictator

[7] "Jabberwacky Live Chat Bot", http://www.jabberwacky.com/

[8] "Stanford Parser", http://nlp.stanford.edu:8080/parser/

CONNECTING HIGH-LEVEL PROGRAMMING CONSTRUCTS TO ASSEMBLY LANGUAGE USING FRANCES*

TUTORIAL PRESENTATION

Kian L. Pokorny, Ph.D.
Division of Computing
McKendree University, 701 College Rd., Lebanon, IL 62254
618-537-6440
klpokorny@McKendree.edu

Tyler Sondag
Dept. of Computer Science
Iowa State University, 226 Atanasoff Hall
Ames, IA 50014
sondag@cs.iastate.edu

Hridesh Rajan
Dept. of Computer Science
Iowa State University, 226 Atanasoff Hall
Ames, IA 50014
hridesh@cs.iastate.edu

Central to computing is machine code generation. Upper level undergraduate students studying computing are quite familiar with high-level languages. Most undergraduate programs in computing begin with a course involving computer programming in a high-level language and as students progress through their studies they gain more experience with high-level languages. Not only is the machine code central to upper level courses in computer science, introductory programming texts address the issue of how high-level languages are translated to execute on the machine [3, 4].

In recent years, as the field of computing has expanded, courses in assembly language have been supplanted with more timely material [1]. In the most recent ACM curricula guidelines assembly language is only mentioned in one of the fourteen sections, Architecture and Organization [2]. One of the ten subsections herein has nine hours recommended and there are nine learning objectives for this subsection. The ninth learning objective is: "Understand how, at the assembly language level, how parameters are passed to subroutines and how local workplace is created and accessed" [2]. Courses in computer architecture, operating systems and compiler design revolve around machine language yet the disconnect that exists from the lack of a course dedicated to the topic of

assembly language introduces a gulf in knowledge. This makes it difficult for students to quickly grasp the concepts of these other courses and requires the instructor to spend time with the topic of assembly language.

Frances [5] is a new tool that shows graphically the relationship between statements in a high-level language and the generated assembly language. This tool allows students to quickly make a connection between the written high-level language and the generated assembly instructions. The environment has several unique features that make it useful to students needing to gain an understanding of this relationship.

Computing faculty teaching courses at all levels will be interested in the potential for classroom use. The presentation will provide instruction on the use of the tool, information about supporting material and access to the tool for future use in the classroom.

REFERENCES

[1] Agarwal, K.K., Agarwal, A., *Do we need a separate assembly language programming course?, Journal of Computing Sciences in Colleges*, 19, (4), 246-251, 2004.

[2] *Computer Science Curriculum 2008: An Interim Revision of CS2001, December 2008.* Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, 2008.

[3] Deitel, P.J., Deitel, H.M., *C++ How to Program 6 ed*, Upper Saddle River, New Jersey: Pearson's Education Inc., 2008.

[4] Gaddis, T., *Starting out with Control Structures Through Objects 6 ed*, Upper Saddle River: Pearson's Education Inc., New Jersey, 2009.

[5] Sondag, T., Pokorny, K.L., Rajan, H., 2010. Frances: A tool for understanding code generation, *Proceedings of the 41st SIGCSE technical symposium on Computer science education, Milwaukee, Wisconsin, USA*, March 10-13, 2010.

AUTOMATED GRADING OF STUDENT PROGRAMMING ASSIGNMENTS*

TUTORIAL PRESENTATION

Stefan Brandle
Computer Science and Engineering
Taylor University
236 W. Reade Avenue
Upland, IN 46989
765-998-4685
sbrandle@cse.taylor.edu

THE ASSESSMENT DILEMMA IN COMPUTER SCIENCE EDUCATION

In a SIGCSE note [1] Henry Walker addresses some of the problems with grading in computer science. He cites a report from the MAA/ACM/IEEE Computer Science Task Force on the Teaching of Computer Science with Mathematics Departments [2] to demonstrate that "grading CS [computer science] takes longer than comparable assignments in mathematics or other sciences; CS grading rates appear more closely tied to foreign language than science. [...] However, while upper-level foreign-language courses often require only 4 or 5 short papers during a semester, many CS faculty believe that one assignment every 2 or 3 weeks provides students with inadequate practice. Thus, in CS, assignment frequency may parallel other sciences, while grading time parallels foreign language." Walker concludes from the task force report that "One can grade approximately one and one-half [times] as many English examinations and almost twice as many calculus examinations in a fixed time period [as computer science]." This is part of the CS assessment burden.

In a special SIGCSE working group report "How Shall We Assess This?" [3] the authors discuss how increased class sizes, distributed learning, and distance learning are forcing changes in current assessment practices. They present the debate over the value of automated assessment. In the paper's introduction section they state:

> "With increasing class sizes in educational establishments worldwide, the practice of assessment is becoming a problematic issue; increased numbers make it more difficult to assess student attainment. If assessments are graded manually, educators must either set fewer assessments tasks or resign

> themselves to a greatly increased marking load. In order to cope with increasing student numbers automated assessment is becoming increasingly important in many courses. The number of papers related to the topic that have been presented at ITiCSE conferences in recent years [...] reflects this increasing interest. Automated assessment can save time and human resources but its adoption must be pedagogically sound. It is a widely held belief that on-line teaching and learning will be the savior of the educational system."

Grading student programs is one of those things that can be a headache for both faculty and students. For faculty – especially if teaching with frequent assignments or larger class sizes – grading takes a lot of time. In addition, it is hard to stay on top of the grading of weekly programs. Students often do not receive feedback quickly enough for it to be helpful for the next assignment and it is frequently not very detailed.

The idea of deputizing the computer to help out with grading is nothing new. J. Hollingsworth's 1960 CACM paper "Automatic Graders for Programming Classes" [4] describes automated grading of punchcard-based programs by a punchcard-based program. Several hundred papers have been published since, but few teachers use automated grading tools. Those who do so, typically use homemade, hard-to-share systems. In order to leverage the ability to share tools, I recently stopped using my homemade system and switched to Web-CAT [5], which supports standard platforms and a range of languages and grading techniques. If one person adds support for a new language or assessment tool, all Web-CAT users can start using it. I am the project contributor of the code that provides Python support.

Attendees will learn about a set of useful tools that are beneficial and reasonably easy for faculty and students to use.

TUTORIAL TOPICS

- Different approaches to automatic program testing. Application testing vs. more detailed unit testing.
- Test-driven development (TDD) vs. non-TDD.
- What is feasible and infeasible (or still hard to do) with automated testing.
- Quick introduction to the xUNIT architecture.
- Examples of how to test code in Python, Java, and C++.
- A demonstration of the Web-CAT automated grading system.

REFERENCES

[1] Walker, Henry M. Notes on grading, *SIGCSE Bulletin, 32 (2)*, 18-19, 2000.

[2] Haytock, B., Karian, Z., Selter, S. Teaching computer science within mathematics departments, *Computer Science Education*, 1, 1991, 1990.

[3] Carter, J., Ala-Mutka, K., Fuller, U., et al. How shall we assess this? *Working group reports, ITiCSE 2003*, 107-123, 2003.

[4] Hollingsworth, J., Automatic graders for programming classes, *Communications of the ACM*, 3 (10), 528-529, 1960.

[5] Edwards, S., Web-CAT, http://web-cat.cs.vt.edu, retrieved January 19 5, 2010.

RESISTOR NETWORK-BASED ALGORITHMS FOR ENUMERATING RATIONAL NUMBERS*

Samuel C. Hsieh, C. Van Nelson, and Logeshbabu Sampath
Computer Science Department
Ball State University
Muncie, Indiana, U.S.A.

ABSTRACT

A class of simple resistor networks is used to enumerate the set of positive rational numbers. A one-to-one correspondence between the simple resistor networks and rational numbers is established, and efficient algorithms to find the rational for a given ordinal number, to find the ordinal number for a given rational, and to generate the rationals up to a given ordinal number are presented. The bijection and the algorithms answer some typical questions that are frequently asked by students in discrete mathematics or computing theory classes related to counting positive rational numbers.

1. INTRODUCTION

The set of positive rational numbers is commonly shown to be countable by using a matrix of fractions [e.g., 1, 2]. Each fraction *i / j* is located in the *i*th row and the *j*th column of such a matrix. An enumeration is then shown by sequentially listing the fractions on the diagonals, starting from the corner where the fraction 1/1 is located, and skipping an element if it causes a repetition. When this approach is used to present the topic in discrete mathematics or computing theory classes, some of the typical questions from the students are:

- What is the mapping from the integers to the rationals?
- Is there an easy way to find the nth rational for a given n?
- Is there an easy way to find the ordinal number of a given rational?

Indeed, the approach constructs a linear list of positive rational numbers but does not provide a formula for the one-to-one correspondence. In particular, the list so constructed

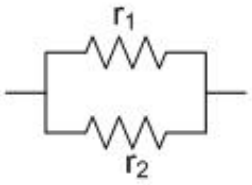
includes "holes" to avoid double counting. The above questions usually arise from the students' concern with such holes in the list and the lack of an explicit formula that avoids double counting.

We will present a mapping that answers such questions. The mapping is based on a class of simple resistor networks and can be defined by simple formulas for computing the equivalent resistance of the networks. Efficient algorithms to find the nth rational for a given ordinal number n, to find the ordinal number for a given rational, and to generate a list of rationals up to a given ordinal number will be presented and analyzed.

Enumerating the rational numbers is a well-studied problem. A salient feature of the approach discussed in this article is that it is based on concrete physical objects: resistor networks.

2. SIMPLE RESISTOR NETWORKS

Let r_1 and r_2 be the resistance values of two resistors. The resistors can be connected (combined) in parallel or in series, as shown below.



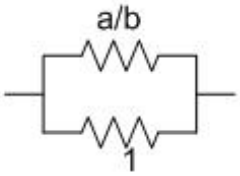
$r_{eq}=(r_1 r_2)/(r_1 + r_2)$

Fig. 1a

$r_{eq} = r_1 + r_2$

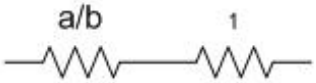
Fig. 1b

When the two resistors are connected in parallel (Fig. 1a), the equivalent resistance (the combined resistance) of the resistor network is $r_1r_2/(r_1+r_2)$. When the resistors are connected in series (Fig. 1b) the equivalent resistance is $r_1 + r_2$.



$r_{eq}=a/(a + b)$

Fig. 2a



$r_{eq} = (a+b)/b$

Fig. 2b

As an example, consider two resistors of resistance a/b and 1 (ohm). The equivalent resistance of the two resistors connected in parallel (Fig. 2a) is (a/b)/(a/b + 1) = a/(a+b), and that of the resistors in series (Fig. 2b) is a/b+1 = (a+b)/b.

We consider a sequence of resistor networks. Each network is constructed from 1-ohm resistors. The first network in the sequence is a 1-ohm resistor. For $n > 1$, if n is even then the n^{th} network is formed from the $(n/2)^{th}$ network by connecting a 1-ohm resistor *in parallel with* the $(n/2)^{th}$ network; if n is odd, then the n^{th} network is constructed from the $(n/2)^{th}$ network (discarding fractional part of the quotient $n/2$) by connecting a 1-ohm resistor *in series with* the $(n/2)^{th}$ network. Fig. 3 shows the first seven networks in the sequence and their equivalent resistance R(1), R(2)..R(7) as fractions.

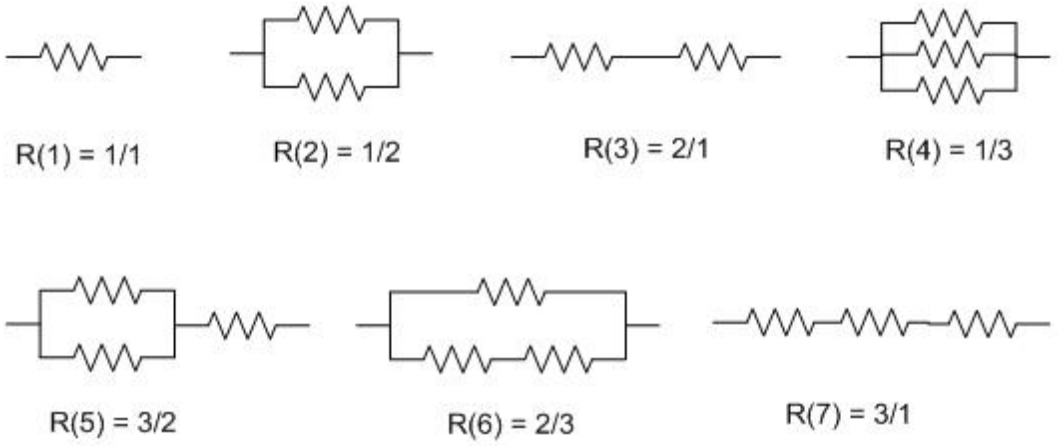


Fig. 3

It should be clear that the 2nd and the 3rd networks are formed from the 1st network, the 4th and the 5th networks are formed from the 2nd network, and the 6th and the 7th networks are formed from the 3rd network. The equivalent resistance of each network can be computed using the simple formulas given in Fig. 2. For example, since the 5th network is the 2nd network with a 1-ohm resistor connected in series, the equivalent resistance R(5) can be computed from R(2) by using the formula in Fig. 2b: since R(2) is 1/2, R(5) is (1+2)/2 = 3/2. As another example, since the 6th network is the 3rd network in parallel with a 1-ohm resistor, the equivalent resistance R(6) is computed from R(3) by using the formula in Fig. 2a: since R(3) is 2/1, R(6) is 2/(2+1)=2/3.

In summary, the equivalent resistance of the first network R(1) is 1/1. To compute the equivalent resistance R(n) of the n^{th} network for any $n > 1$, let R(n/2) be a/b (if n is odd, the fractional part of the quotient n/2 is discarded). R(n) is a/(a+b) if n is even and (a+b)/b if n is odd. The next section will establish that R is a one-to-one correspondence between the set of positive integers and the set of positive rationals. Hence, the set

$$\{ R(n) \mid n \geq 1 \}$$

is the set of all positive reduced fractions, i.e., the set of all positive rationals, and the

sequence R(1), R(2), R(3)… is an enumeration of them.

3. ONE-TO-ONE CORRESPONDENCE

For any positive integer k, R(k) is a reduced fraction because

- R(1), i.e., 1/1, is reduced, and
- Any fraction of the form (a+b)/a or the form a/(a+b) derived from a reduced fraction a/b is also reduced.

Suppose R(k) is a/b for some positive integer k. We note, from the definition of R, that

- a=b if $k = 1$, and
- a > b if k is odd and $k \neq 1$, and
- a < b if k is even.

We will now show, by contradiction, that R is a one-to-one correspondence. That is, R is both injective (one-to-one) and surjective (onto).

Suppose R is not injective. There exist two positive integers j and k such that $j \neq k$ but R(j) = R(k). Without loss of generality, let's assume $j < k$. Since R(j) = R(k), j and k must either be both odd or both even. Since R(j) and R(k) are derived from R(j/2) and R(k/2) in the same way, R(j/2) must be the same fraction as R(k/2) and, therefore, j/2 and k/2 must either be both odd or both even. Besides, j/2 < k/2 because $j < k$ and j and k are either both odd or both even. For the same reasons, the invariant that R(j) = R(k), j and k are either both odd or both even, and $j < k$ holds for the values of j and k right after every iteration of the following loop in C++:

```
while (j != 1)
{j = j/2;
 k = k/2;
 //at this point R(j) = R(k), j and k are either both be odd or both even, and j < k
}
```

When $j = 1$, the loop invariant becomes a contradiction because R(1)=1/1 but R(k) cannot be 1/1 since $1 < k$. Therefore, R is injective.

Now suppose R is not surjective. There exists a positive reduced fraction a/b that is not in the range of R. If a < b, then a/(b-a) is not in the range of R - otherwise, there would be a positive integer n such that R(n)= a/(b-a), and a/b would be in the range of R because R(2n) would be a/b. Similarly, if a > b, then (a-b)/b cannot be in the range of R. By the same reasoning, the invariant that a/b is not in the range of R holds for the values of a and b right after every iteration of the following loop in C++:

```
while (a != b)
{ if (a > b) a=a-b;
  else       b=b-a;
//at this point, the fraction a/b is not in the range of R
}
```

We note that the above loop implements the Euclidean algorithm for computing the greatest common divisor of a and b. Since a and b are initially coprime (because a/b is a reduced fraction), their greatest common divisor is 1. When a=b=1, the loop invariant becomes a contradiction because 1/1 is in the range of R. Therefore, R is surjective.

3. ALGORITHMS

In the algorithm given below, a fraction `num/den` will be represented by the following structure in the language C++:

```
struct Fraction
{ int num;  //numerator of a fraction
  int den;  //denominator of a fraction
};
```

An algorithm to compute R(*n*) for any given n is presented below as a function `findR` in the language C++. The function takes an input parameter `n` and returns a fraction. The algorithm closely follows the definition of R presented in the previous section.

```
Fraction findR (int n)
{ Fraction result;
  if (n == 1)
     { result.num = result.den =1;
     }
  else
     { result=findR(n/2);
       if (n%2) result.num += result.den;
       else     result.den += result.num;
     }
  return result;
} //end function findR
```

Since both `n` and 2 are of the `int` type, the division yields an `int` quotient (e.g., 7/2 is 3). Since every recursive call to `findR` reduces the parameter by at least one half, the running time is obviously O(log *n*).

As an example, let's consider the function call `findR(5)`, which will compute the fraction `R(5)=3/2`. The following calling chain will take place: `findR(5)` calls `findR(2)`, which calls `findR(1)`. Then,

`findR(1)` returns `1/1`, which is used in
`findR(2)` to compute and return `1/2`, which is used in
`findR(5)` to compute and return `3/2`

Since the mapping R is a one-to-one correspondence, for any given rational a/b, its ordinal number *n* can be computed such that R(*n*) is a/b. The algorithm to do so is presented as a C++ function below. The function `findOrd`, when given a reduced fraction a/b as two integers via the parameters, returns the ordinal number of a/b.

```
int findOrd (Fraction f)
{
  if (f.num < f.den)
      { f.den -= f.num;
        return findOrd(f) *2;
      }
  else if (f.num > f.den)
      { f.num -= f.den;
        return findOrd(f) *2 + 1;
      }
  else return 1;
```

```
}
```

In terms of the ordinal number *n*, the algorithm is O(log *n*) time. In terms of the parameters `a` and `b`, the algorithm is linear-time with the worst case running time being O(max(`a, b`)).

Clearly, the function `findOrd` performs the inverse of what `findR` does. As an example, the function call `findOrd(3/2)` will call `findOrd(1/2)`, which will call `findOrd(1/1)`. Then,

`findOrd(1/1)` returns 1, which is used in
`findOrd(1/2)` to compute and return 1*2=2, which is used in
`findOrd(3/2)` to compute and return 2*2+1=5

Note the calls to `findOrd` and `findR` involve the same sequence of rationals (1/1, 1/2, 3/2) and of ordinal numbers (1, 2, 5).

The C++ function `enumerate`, given below, returns a sequence of rationals R(1), R(2), ..R(*n*) for any given *n* in an array `A` of fractions. The array `A` will contain reduced fractions such that `A[i]`is R(`i`). To call this function, the actual parameter for the array `A` must be of size *n*+1 or larger. For ease of presentation, the first element of the array `A` is left unused.

```
void enumerate(int n, Fraction A[])
{  int i;
   A[1].num=A[1].den=1;
   for (i=2; i <= n; i++)
     { A[i] = A[i/2];
       if (i%2)
           A[i].num += A[i].den;
       else
           A[i].den += A[i].num;
      }
}
```

The main control structure of the function is a loop that derives `A[i]` from `A[i/2]` for all `i` in the range `2..n` in accordance with the definition of R. The running time is O(*n*).

4. SUMMARY

An enumeration of the positive rationals based on a sequence of resistor networks is presented. Efficient algorithms to find the rational for a given ordinal number, to find the ordinal number for a given rational, and to generate the rationals up to a given ordinal number are presented and analyzed. The enumeration and algorithms answer some typical questions frequently asked by students in discrete mathematics or computing theory classes.

REFERENCES

[1] Rosen, K. H., Discrete Mathematics and Its Applications, McGraw-Hill, 6th ed., 2007.

[2] Sipser, M., Introduction to the Theory of Computation, Thomson Course Technology, 2nd ed., 2006.

A WEB SERVICE MODEL FOR CONDUCTING RESEARCH IN IMAGE PROCESSING*

Chengcheng Li, Ph.D.
Department of Technology Systems
East Carolina University
Greenville, NC, USA
liche@ecu.edu

ABSTRACT

This study is an online application development process, covering a range of computing science topics, including image processing, feature extract, statistical analysis and artificial intelligence, .NET programming, Matlab programming, web development, and database design and implementation. An online cattle meat quality assessment model is created. Users can contribute and share ultrasound scans of live cattle through the image database management system. Real-time meat quality assessment is performed on the server side using Matlab .NET components. The algorithms are based on linear regression analysis and neural networks. The algorithms yield high accuracies in predicting the intramuscular fat (IMF), which is an important indicator of cattle meat quality. By reading the meat qualify predication from this online approach, the farmers can know the values of the cattle and adjust the feeding method accordingly to maximize profit.

BACKGROUND

The research on beef quality grading through ultrasound technology started more than a decade ago [1][2]. The low resolution and high noise-to-signal ratio of ultrasound images and the limited computing power of the early studies resulted in low accuracies of IMF prediction. In fact, the technician visual appraisal of beef IMF on carcass is still the primary means to evaluate the beef quality today. Stimulated by contemporary genetic

research, computer based IMF assessment on live cattle has yielded encouraging results [3][4]. However, the technology is still limited in the laboratory for research purpose. There are few commercial software products for IMF assessment on the market and they have been used in only testing areas across the country due to the limited available ultrasound scan data and lack of standardized image processing and AI predication algorithms in this field.

Computer-based IMF assessment on farmed animals is very difficult because of the complicated data collection process, which leads to big data variances. Meat quality can be determined by characters other than those shown on the ultrasound images. These characters include gender, age, weight, and breed. Research on cattle feeding and genetic optimization, consumer's satisfaction based on IMF percentage [5], and nutrition values have been the major force propelling the improvement of cattle meat quality and productivity. There is a strong need for real-time live cattle IMF monitoring, so the researchers can adjust their research process accordingly. The farmers will also welcome the live cattle IMF percentage reading technology. They can know the exact value of the cattle before the cows are sent to the slaughterhouse. The farmers can adjust their feeding methods and find the best time for slaughtering, maximizing the value the cattle [6].

INTRODUCTION

The goal of this study is to produce a prediction model on live cattle IMF percentage. There are three principle aims that have been achieved in the study.

1. Develop Prediction Algorithms

A feasible algorithm to predict IMF percentage based on features extracted from live cow ultrasound images is produced. Classification methods such as linear regression analysis and neural networks are used and compared to train the algorithm. The method, which produces the best results in terms of computing time and accuracy, is chosen as the optimal method to predict IMF percentage. The goal of the algorithm is to provide meat quality assessment with accuracy high enough to be approved by the USDA.

2. Build a Live Cow Ultrasound Image Database

Establishing an online publicly accessible database of live cattle ultrasound images one the objectives of this study. The database is used as the primary venue for the online processing and storage of images. The database provides researchers a friendly interface for uploading and querying images, so that the researchers could develop and implement new methodologies to predict IMF percentage on the given images. The database can also be used as a common platform to standardize the format of cow ultrasound images obtained from different resources. An online publicly accessible solution inevitably stimulates collaboration of research on the live cow ultrasound technology usage throughout the country. Input from the same discipline is highly expected.

3. Develop an Online Image Processing Software Application

An executable software application is produced. This application provides a friendly user interface to display the ultrasound images and a clear output of the predicted IMF percentage value. An interactive design allows the users to input and modify data which are not showing on the image, such as the sex, the weight, and the age of the cow. This

software application works with the online database system on the background. It adds new entries to the database based on the user's input. This application is written in ASP.NET and should be executable within the major web browsers such as IE and Firefox under various operating systems.

METHODOLOGY

1. Algorithm Development

This study continues a previous study [7] and investigates different classification methods in linear multilinear regression analysis and neural networks. Trained technicians are used to collect ultrasound images of live cattle from Brown Loam Experiment Station (BLES). The cows are constrained by a specially made machine available at BLES, so that they could be held very still to reduce the noise in the ultrasound images. Aloka 500V SSD ultrasound machine is used to collect high resolution grey-scale images on 11th to 14th ribs of the cattle. Four images are collected on each cow. The cows are sent to the slaughterhouse. The USDA (United States Department of Agriculture) certified technicians visually grade the carcass' marbling levels. Chemical analysis is conducted to produce detailed quantitative data of the IMF percentage. The correlation between live images and the fat percentage data is investigated through a series of image processing operations and statistical analyses, leading to a highly accurate and the USDA approved IMF percentage prediction model on live cows.

384 ultrasound images were scanned from 91 live steers to train and test the proposed algorithms. The region of interest (ROI) is automatically extracted based on the image features. The training-testing ratio is 3:1 which means seventy five percent of the images are used for the algorithm training purpose while the remaining images are used to test the prediction accuracy of the algorithm.

A large number of image processing features are extracted from the ultrasound scans. This number is reduced according to the correlation between the individual features and the prediction results. It is important to reduce the dimensionality to eliminate the uncorrelated features and make the real-time calculation possible. After sufficient training, the algorithms generated from linear regression analysis and neural networks both produce very high accuracy in predicting the IMF value. The following table compares the effectiveness of the algorithms from this study and previous ones.

TABLE 1. Quantitative evaluation of IMF prediction algorithms

	Whittake [2]	Hanssen [8]	Brethour [9]	**Li (LRA)[7]**	**Li(NN)[7]**
Accuracy	41% - 70.8%	N/A	74%-82%	**92.5%**	**89.8%**
R^2	0.37-0.82	0.69-0.72	0.32	**0.823**	**0.809**
RMSE	0.503-2.781	0.84-0.91	N/A	**0.196**	**0.253**

The accuracy is calculated based on

$$\text{Accuracy} = 1.0 - [(|\text{Estimated\_Value} - \text{True\_Value}|)/\text{True\_Value}] \quad (1)$$

R2 and RMSE (root of mean squared error) are also used to compare the algorithms.

2. Online Image Repository System and Meat Quality Assessment

An image database is setup for researchers and farmers to upload cattle ultrasound scans to a central online site. This online solution helps researchers to gather and share data through the internet so that a large training set of data can be obtained. The data is archived in an MS SQL Sever. The database interfaces the users through .NET technology. A friendly user input page is developed as the follows.

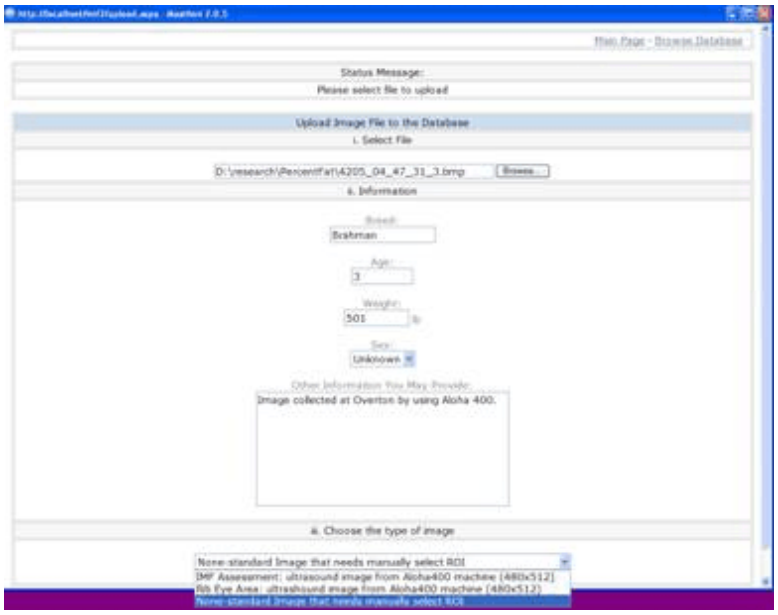


Figure 1. User upload page.

The user can upload an image from local disk to the online database. Other information accompanying the image can be also input. The images are classified as IMF analysis, rib eye area size assessment, or just non-standard scan. For IMF and rib eye scans from standard collection process using Aloha ultrasound machines and with an image size of 480X512 pixels, the current system can automatically extract the ROI (region of interest) from the image. Otherwise, the user has to determine the ROI area. The image operation page, shown in Figure 2, allows user to further operate the images within the database. The user can view the image by clicking the image name, delete or download an image to the local disk through the "Del" and "Dnld" options, and read the IMF and rib eye size prediction.

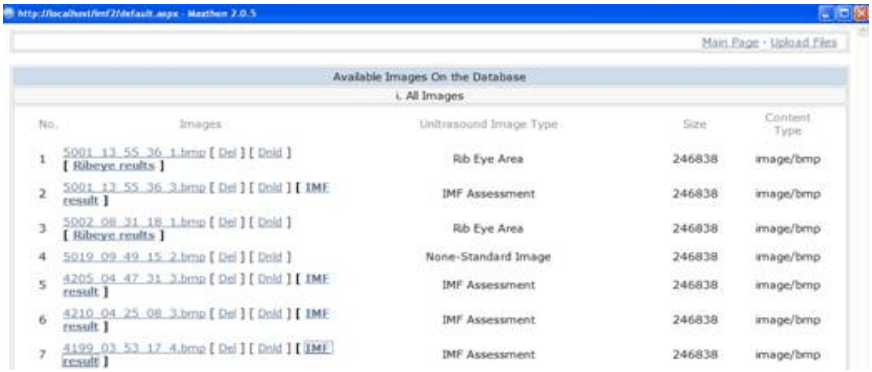


Figure 2. Image operation page

Matlab is a very convenient computing environment when it comes to image processing and matrix calculation. The embedded Matlab image processing and statistics toolboxes are used to train the algorithms. Matlab objects or components are created and placed on the server side so that they can be referenced by C# codes which are used to create the website and the program-behind-the-page. The MATLAB.NET.Arrays.MWArray component is particularly important for calculating the image information from the matrix-like grey-scale pixel values. The combination of Matlab component and the .NET technology results in fast real-time calculation and relatively easy application development.

Figure 3. Viewing images

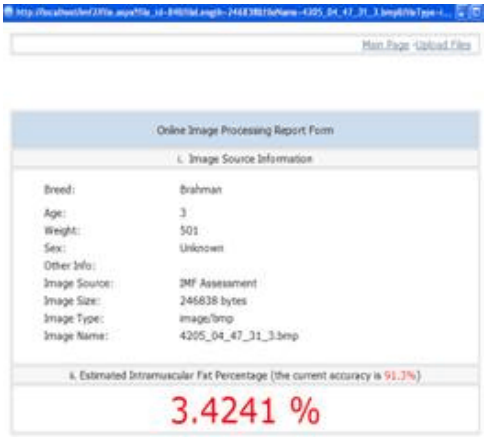


Figure 4. Assessment report

Figure 3. shows an sample image stored in the database. User can view the image by clicking the file name from the image operation page. An 80X80 ROI region is automatically extracted from the image. The analysis of the ROI region is performed on the background through the interaction between the C# programming and the Matlab components. The analysis result can be viewed by clicking the "IMF result" or "Ribeye result" hyperlinks on the image operation page. A sample of IMF analysis report is shown in Figure 4. It shows the information of the cow that was input by user and an estimated IMF value of 3.4%.

CONCLUSIONS

The study is a complete online application development process, covering a wide range of computing science topics, including image processing, feature extract, statistical analysis and artificial intelligence, .NET programming, Matlab programming, web development, and database design and implementation. It can work as a team project for CS major students.

The online meat quality assessment site, the deliverable of this study, has great practical values. It creates a common platform, provides data, and establishes standards for the researchers to conduct research in animal science through image processing. This model can be easily adopted and disseminated to other fields.

ACKNOWLEDGEMENT

This study is supported by a research project that is funded by ECU College of Science and Technology.

REFERENCES

[1] James D. Mccauley, Brian R. Thane, A. Dale Whittaker, "Fat Estimation in Beef Ultrasound Images Using Texture and Adaptive Logic Networks" Transaction of ASAE, 1994, Volume 37, pp. 997-1002

[2] Whittaker, A. D., B. Park, B. R. Thance, R. K. Miller, and J. W. Savell, "Principles of Ultrasound and measurement of Intramuscular Fat", Journal of Animal Science, 1992 volume 70, pp. 942-950

[3] Tait Jr., D.E. Wilson and G.H. Rouse, "Prediction of retail product and trimmable fat yields from the four primal cuts in beef cattle using ultrasound or carcass data", Journal of Animal Science, 2005, pp. 1353-1360.

[4] A. R. Williams, "Ultrasound applications in beef cattle carcass research and management", Journal of Animal Science, 2002, Volume 80, pp. 183-188

[5] Fernandez X., Monin G.,Talmant A., Mourot J., Lebret B., "Influence of intramuscular fat content on the quality of pig meat", Journal of Meat Science, volume 53, number 1, September 1999, pp. 67-72

[6] Corino C., Magni S., Pagliarini E., Rossi R., Pastorelli G., Chiesa L.M., "Effects of dietary fats on meat quality and sensory characteristics of heavy pig loins", Journal of Meat Science, Volume 60, Number 1, January 2002, pp.1-8

[7] Chengcheng Li, Yufeng Zheng, Kwabena Agyepong, "Prediction of IMF Percentage of Live Cattle by Using Ultrasound Technologies with High Accuracies", ICIE '09. WASE International Conference on Information Engineering, July 2009, Volume 2, pp. 474 - 478

[8] A. Hanssen, D.E. Wilson, R.L. Wilham, G.H. Rouse and A. Trenkle, "Evaluation of ultrasound measurements of fat thickness and longissimus muscles area in feedlot cattle: Assessment of accuracy and repeatability", Canadian Journal of Animal Science, 1998, pp. 227-285.

[9] R. Brethour, "Using receiver operating characteristic analysis to evaluate the accuracy in predicting future quality grade from ultrasound marbling estimates on beef calves", Journal of Animal Science, 2000, volume 78, pp. 2263-2268

WRONG NUMBER: AVOIDING THE HIDDEN PERILS IN IPHONE DEVELOPMENT*

Michael P. Rogers
Computer Science Information Systems
Northwest Missouri State University
Maryville, MO 64468
Michael@nwmissouri.edu

ABSTRACT

Apple's iPhone represents an exciting opportunity for student developers to hone their skills on a compelling platform. However, developing for the iPhone is also extremely challenging, and faculty interested in guiding student developers should be aware of some potential pitfalls. This paper will detail some of the most common difficulties encountered in iPhone development, and strategies for avoiding them.

1. INTRODUCTION

The iPhone almost needs no introduction. On the basis of an elegant user interface and superb engineering, it has, in just over 2 years, risen from obscurity to become one of the world's best selling smartphones, with total sales of over 30 million units. The iPhone was originally envisioned as a closed platform; but after a public outcry, Apple opened up the device to 3rd party developers. More than 85,000 applications ("apps") have been written for the device, and there have been more than 2 billion downloads from The App Store, which Apple uses to distribute iPhone apps.

These are astonishing numbers, but because the iPhone's rise has been so meteoric, the materials that typically emerge to explain a new technology are still scarce. Good texts on iPhone development are now just starting to appear. Most faculty have no experience with the device, which presents problems when students approach, as they surely will, asking for perhaps an independent study or even a course on the subject.

This paper attempts to help remedy that situation, by describing the most common

---

pitfalls encountered in iPhone development and how to avoid them.

2. MASTERING THE USER INTERFACE

The first, and what in retrospect should be the least surprising pitfall, is that while students might - for reasons alluded to above - be eager to do iPhone development, they might not have any actual experience with iPhones. And this applies to more than just students: it is possible to envision a scenario where, in a capstone course with an outside client wanting an iPhone app, *none* of the other participants have actually used an iPhone. Needless to say, this could be a recipe for disaster.

The trouble is that, while the iPhone is easy to use, its user interface is nonetheless novel; even the physical motions that one makes when using the iPhone are different than on a computer. Consequently, it is essential that would-be developers spend *hands-on* time using an iPhone: a mentoring faculty member should demonstrate and then turn over the device for the students to explore, to develop a visceral feel for its behavior.

That exploration should start with the apps that ship with the iPhone. They are written by Apple, and exercise all of the standard interface paradigms: so they may be considered definitive, working models of how an iPhone app should behave. Students also should read Apple's thoughtful and comprehensive iPhone Human Interface Guidelines [1].

In many environments, an excellent way to test the user interface is with a prototype. However, a curious point about iPhone development is that making a functional, or even accurate-looking prototype, requires coding. This is because Apple's engineers adhere closely to the model-view-controller design pattern. Views (technically, UIViews, the highest level object in the hierarchy with a physical representation on the screen, and their myriad subclasses) really *are* just something to look at. Navigating from one UIView to another, or even filling a data view (e.g., populating a table, or placing a locator pin on a map) requires writing a controller.

This has implications for iPhone neophytes starting a larger, perhaps semester-long project. They really will want a working prototype to test their design, as there is a good chance that they will get at least parts of it wrong the first time. Lacking the coding skills to make one, they should be encouraged to resort to paper: 3x5 index cards, or even 2 judiciously fastened sticky notes, are about the right aspect ratio to represent an iPhone.

3. OBTAINING HARDWARE

Development for the iPhone does not actually require an iPhone. The iPhone Software Developer's Kit (SDK) ships with a simulator that mimics nearly all of the iPhone's functionality, lacking only a virtual camera and GPS. In addition, the SDK includes Cocoa Touch, the frameworks on which iPhone applications are based; Xcode, a powerful IDE which can be also be used to develop Mac OS X applications (iTunes, Safari, and Mac OS X itself were written with it); Interface Builder, for designing the user interface; and, finally, Instruments, a sophisticated profiling tool that can provide insight into how an app is managing (or mismanaging) memory and other resources. The iPhone SDK only runs on recent vintage Macintosh computers. However, the iPhone SDK itself

is relatively resource friendly, so *any* recent vintage Mac should suffice.

At some point, students will need to run their app on an actual phone. Not only is it the endgame of iPhone development, but also there are simply some aspects of usability (most notably those involving the touch screen) that cannot be evaluated without one.

Making iPhone ownership a prerequisite for enrolling in a course would certainly raise eyebrows, and departmental purchases are also problematic: iPhones require a 2-year contract with AT&T, and potentially variable monthly bills might prove awkward.

A possible alternative might be to use an iPod Touch - a device that has many of the iPhone's features, and is code-compatible, in that an app written for an iPod Touch will work on an iPhone. The iPod Touch does lack a camera and GPS, but other software will run exactly as on an iPhone.

4. COPING WITH A NOVEL OPERATING SYSTEMS AND FRAMEWORKS

The iPhone runs a modified version of Mac OS X. Consequently, students who have written Mac OS X programs, using the *Cocoa* frameworks, will have an advantage when writing iPhone apps using the aforementioned Cocoa Touch. They share the *Foundation* classes - those lacking a visual presence, such as data structures (NSDictionary, NSArray), common data types (NSString, NSNumber), etc. At the GUI level they do diverge, with Cocoa's AppKit and Cocoa Touch's UIKit offering classes tailored to their respective platforms. Yet even here there is considerable overlap in nomenclature and concepts.

However, the reality is that the number of students able to leverage their Mac OS X development experience is small: indeed, at most universities, Computer Science students will not have even seen Macintoshes in a classroom setting. So, while Mac OS X does have a reputation as being easy to use, students will still benefit enormously from an orientation session covering system configuration and administration, file system navigation, etc.

Turning to Cocoa Touch itself, student developers will find themselves in a situation that faculty who have been teaching for any length of time are already familiar with: the need to assimilate and become productive in a new environment. It will require, in this case, acquiring a new language and concepts, and learning new design patterns. While undoubtedly challenging, this is excellent preparation for the workforce. It is axiomatic that students will have to do the same after they graduate, and what better place to practice than in a controlled, low-pressure environment with a sympathetic mentor?

It helps that Cocoa Touch is based on a venerable, popular, and well-documented framework: Cocoa has been in existence in one form or another since the 1980's. However, there are some technical issues that will trip up the unwary: we turn to those now.

5. CHOOSING THE BEST LANGUAGE

Most iPhone applications are written using Objective-C, a variant of ANSI C with object-oriented extensions based on the syntax of Smalltalk. Objective-C is a relatively

easy-to-learn language, especially since most of the awkward bits of C - arrays, strings, and pointers - are almost never seen in a standard iPhone app. Instead, coders use NSArrays and NSStrings; pointers are typically required only to declare an object, and never explicitly dereferenced. While the syntax for method calls is different - parameters are labeled, and object messaging uses square brackets rather than dot notation - diligent students quickly become proficient in the language.

Figure 1 shows the interface section of a class named Submarine. Objective-C classes are divided into interfaces and implementations, and usually stored in separate .h and .m files, respectively. The interface section encloses its variables in curly brackets, followed by the definition of its constructors, methods, and properties. The compiler directives @interface and @end mark the beginning and end of the interface. The implementation section, not shown here, is similarly defined between @implementation and @end directives.

```
@interface Submarine {
    bool periscopeUp; // instance variables are defined in {}
                  // section
    double depth;
}
-(id)init;           // constructor, returns id (a generic type)
+(Submarine *)createSub; // class (+) method that returns a
                    // Submarine *
-(void)raisePeriscope;   // an instance method with no parameters

-(void)diveToDepth:(int)depth withAlacrity:(bool)alacrity;
@end                 // diveToDepth:withAlacrity has 2 parameters
```

Figure 1

Using the Submarine class is quite easy, as shown in Figure 2.

```
Submarine * nautilus =  [[Submarine alloc]init];

[nautilus raisePeriscope];    // sending a message (raisePeriscope)
                              // to an object (nautilus)

[nautilus diveToDepth:500 withAlacrity:YES];
```

Figure 2

In the first line of Figure 2, we send a class method, alloc, to the Submarine class, to allocate memory for the object; the object that is returned is then init'd (that is, we call its constructor). The careful reader may have noticed that init returns a data type called id; this is a generic type, capable of storing any reference object. We could have had it return a Submarine pointer, but id is more traditional. In the second line, we send a raisePeriscope message to nautilus, and in the third line, we see the advantage of labeling parameters. An equivalent Java statement:

```
nautilus.diveToDepth(500,true);
```

might have had the reader puzzling as to what the second argument signifies, in contrast to the Objective-C code, where it is explicitly spelled out.

The danger, when it comes to languages, is in adopting one *other* than Objective-C. Apple provides Cocoa bindings for Ruby and Python; Mono promises a tie to C#; and

there are other beguiling alternatives. However, all of Apple's documentation and virtually all third party texts use Objective-C. It is also the language of choice on most online forums. In short, the few hours required to master Objective-C will pay big dividends in the long run.

6. GETTING A HANDLE ON MEMORY MANAGEMENT

One of the most confusing aspects of iPhone development is memory management. While garbage collection is available for Mac OS X development, it is not supported on the iPhone: the developer has sole responsibility for allocating and releasing memory according to a set of clearly defined rules.

Briefly, any object that is allocated must, at some later date, be released; so for example, each statement that allocates an object:

```
Submarine * nautilus = [[Submarine alloc] init];
```

must, at some later stage, be balanced by a statement that releases it:

```
[nautilus release];  // memory for nautilus is now freed up
```

So far, this is straightforward. But what about objects that are instantiated in a method and then returned? For instance, suppose that nautilus is instantiated in a class method called **createSub**, in a class called Submarine, and then returned.

```
+(Submarine *) createSub {
    Submarine * nautilus = [[Submarine alloc] init];
    return nautilus;
}
```

Once createSub finishes, nautilus goes out of scope, making it impossible to send it a release message. Each time createSub is invoked, it would leak a Submarine.

Apple's solution is to add these objects to an *autorelease* pool. As the name suggests, objects in an autorelease pool will be automatically released, shortly after the method that created them has completed. If other objects wish to use autoreleased objects, they may: but if they need them for an extended period of time, they must remove them from the autorelease pool by sending them a **retain** message. They effectively take ownership of the autoreleased objects, and so owe them a release message when finished with them.

These complex rules become clear with practice, and several recent texts [2, 4] do a good job of explaining them. If all else fails, memory leaks are easy to spot using Instruments.

7. UNDERSTANDING THE OUTLET DESIGN PATTERN

Easily the most confusing design pattern, for beginning iPhone developers, is that of Outlets. When developing controllers and views, controllers are written in Xcode, in standard .h and .m files. But the views are laid out using Interface Builder, in what are called .xib files. Somehow, programmers must be able to connect the two, so that controllers can access/manipulate the GUI widgets on the view.

The solution [2] relies on two main ideas. First, the .xib file contains not just a

view, but also a *proxy* for the controller. Secondly, the controller adorns some of its variables with an empty C macro, IBOutlet.

Variables so adorned become visible (or exposed), within Interface Builder, as outlets: the programmer physically drags a line from a proxy's outlet to a chosen GUI widget and in doing so, establishes a connection. From then on, any time the controller works with the outlet variable, it will in fact be working directly with the GUI widget.

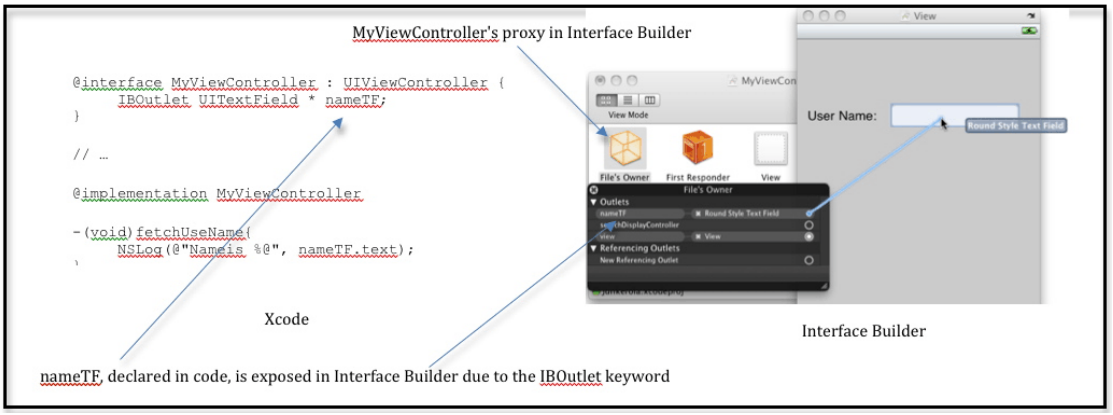


Figure 3

8. SUMMARY

iPhone development, although rewarding and motivating, is fraught with peril. Students will be challenged by the novelty of the language, frameworks, operating system, memory management and design patterns. If given proper guidance, however, they may very well exceed in this exciting new arena; and by following the advice described in this paper, they will be more likely to do so.

9. REFERENCES

[1] Apple, Inc. iPhone Human Interface Guidelines, developer.apple.com/iphone, retrieved December 9, 2009.

[2] Buck, Erik M. and Donald A. Yacktman. *Cocoa Design Patterns*. Addison-Wesley, 2009.

[3] Mark, Dave and Jeff LaMarche. *Beginning iPhone Development Exploring the iPhone SDK*. APress, 2008.

[4] Sadun, Erica. *The iPhone Developer's Cookbook: Building Applications with the iPhone 3.0 SDK*, Second Edition. Addison-Wesley, 2009.

PROGRAMMING USER INTERFACES USING THE NINTENDO WII REMOTE*

TUTORIAL PRESENTATION

Chuck Pheatt
Department of Computer Science
Emporia State University
Emporia, KS 66801
620 341-5637
cpheatt@emporia.edu

Scott Goering
Department of Computer Science
Emporia State University
Emporia, KS 66801
620 341-5281
sgoering@emporia.edu

ABSTRACT

The NintendoWii gaming console utilizes a 3D user interface device called the Wii Remote, also known as the Wiimote. The Wiimote incorporates a number of buttons for console communication as well as an accelerometer and IR sensor. In addition, the device allows accessories to be attached (such as the Classic Controller and Nunchuck) to extend its capabilities. The Wiimote utilizes the Bluetooth wireless protocol for communicating with the game console.

The Wiimote has been used in a wide variety of applications beyond simply acting as a game console interface. Applications range from the seminal work with the Wiimote device by a Carnegie Mellon University researcher involving an interactive whiteboard [1] to the control of a 15 ton construction grapple by an Australian engineering firm [2]. It has become the favorite device with homebrew application developers [3] since interfacing the Wiimote to a PC simply requires a low cost Bluetooth dongle (<$20).

In this tutorial we will present an easy to use software framework for utilizing the WiiMote as a PC input device. Building on an existing library [4], we've developed a series of examples that illustrate how the Wiimote may be used as a sophisticated input device. The goal is to provide examples that can be used and extended by students without detailed knowledge of user interface messaging requirements.

Examples included in the presentation will be:

- Basic Wiimote and Bluetooth setup and "Hello World" applications.
- Understanding of control button and sensor inputs including the accelerometer and

IR sensors.
- Integrating input from external controls including the Classic Controller and Nunchuck.
- Developing a classic game using simple sprites.
- Developing a gaming environment using OpenGL.

Programming examples may be utilized by introductory programming students as well as computer science undergraduates developing unique user interfaces or pursuing an independent study.

REFERENCES

[1] Lee, J., Hudson, S., Summet, J., and Dietz, P. "Moveable Interactive Projected Displays Using Projector Based Tracking", *Proceedings of the ACM Symposium on User Interface Software and Technology*, October 2005. pp. 63-72.

[2] Transmin Engineering, http://www.transmin.com.au. Video appears at http://inventorspot.com/articles/tiny_wii_device_controls_giant_scary_grappling_machine_31009

[3] Hack a Wii, http://hackawii.com.

[4] Brian Peek, Managed Library for Nintendo's Wiimote, MSDN Coding for Fun, http://blogs.msdn.com/coding4fun/archive/2007/03/14/1879033.aspx.

TEACHING NETWORKING AND DISTRIBUTED SYSTEMS WITH SEATTLE*

TUTORIAL PRESENTATION

Justin Cappos, Ivan Beschastnikh
Department of Computer Science and Engineering
University of Washington
justinc@cs.washington.edu, ivan@cs.washington.edu

ABSTRACT

In this tutorial we describe an educational testbed called Seattle [1] that an instructor can use to enhance a course on networks or distributed systems. We give an overview of the Seattle educational testbed and describe assignments and resources available to educators planning to use Seattle in the classroom. Finally, we discuss the applicability of Seattle to other types of distributed systems paradigms such as cloud computing, peer-to-peer networking, and ubiquitous computing.

The purpose of the Seattle testbed is to provide instructors with resources on computers all around the world. This allows students to learn about real world network behavior as well as topics like cloud computing and peer-to-peer computing. Seattle is free and is comprised of resources donated from universities. Today, Seattle consists of resources on about 1000 computers at over 100 universities on 6 continents.

OVERVIEW OF SEATTLE

Seattle is a community-driven testbed of computers from around the world. It is completely free for anyone to use. Universities and individuals donate available computational resources on multi-user machines, such as the machines in a computer laboratory. Resources are donated by installing Seattle software, which runs in the background while the computer is being used for other tasks. Seattle uses incentives to drive adoption of the testbed. As of December 2009, for every computer running the donation software, the testbed provides resources on ten other computers to the person making the donation.

Seattle is safe to install because it has protection built-in (however may be installed in a restricted account if desired). Seattle provides protection through a sandboxed environment that provides strict isolation guarantees and resource restrictions. This prevents a student program from impacting the security or performance of the donating computer. Seattle programs are written in a specific language (a restricted version of Python). This makes it possible to ensure the safety of the programs.

Students can run Seattle programs by using a shell called seash. To run the student's code on a remote computer, the shell contacts a program called the node manager that runs on the remote computer. The node manager also ensures that the total amount of resources consumed by all programs is kept below 10%. As a result, the performance impact of Seattle on other users of the donated computer is minimal.

The Seattle software requires no maintenance or overhead for the university's technical support staff. Seattle software comes with a software updater, which automatically keeps the Seattle software up-to-date without any user or administrator intervention.

The SeattleGENI website facilitates sharing of resources [2]. This web site allows instructors to share the donation credits they receive with students taking their courses. It also allows students to request resources on computers with specific network characteristics, such as LAN or WAN.

The Seattle website contains educational materials including the student portal and the educator portal. The student portal contains tutorials that introduce students to Python, teach the restricted Python language used to write Seattle programs, explain how to run Seattle programs locally, and describe how to use seash for remote execution.

The educator portal contains a set of tested assignments for instructor including a complete semester's worth of assignments for both undergraduate and graduate networking classes. For example, the Take-Home Assignment teaches students about connectivity on the Internet. The goal is to teach student about non-transitive connectivity and NATs (Network Address Translation) boxes. The student locates non-transitive connectivity and sets up one-hop routing to avoid it. The student also determines if their computer is behind a NAT. This is a good first assignment for a class because it only takes an hour to complete and requires no programming.

TUTORIAL TOPICS

In this tutorial, we will show instructors teaching networking and distributed systems courses the power of using the Seattle testbed. This will include:

1. An overview of the instructional resources available for Seattle including software, documentation, and course assignments.
2. A brief overview of how to install and configure Seattle.
3. A demonstration of how an instructor might use Seattle in a classroom with a live programming session of an existing assignment.

REFERENCES

[1] "Seattle: The Internet as a Testbed." Accessed December 14th, 2009 at https://seattle.cs.washington.edu/

[2] "GENI | Global Environment for Network Innovations using Seattle." Accessed March 12, 2009 at https://seattle.cs.washington.edu/geni/

EXPERIENCES WITH VIRTUALIZAION TECHNOLOGY IN EDUCATION*

Timothy Bower
Kansas State University at Salina
785-826-2920
tim@ksu.edu

ABSTRACT

This paper describes how virtualization technology was used to facilitate the teaching of three classes and presents heuristic results describing when certain technologies were preferred over others.

INTRODUCTION

Virtualization technology has relevance to the class room as well as to the server room. Software virtualization technology is currently one of the most frequently discussed topics in information technology (IT) and systems administration trade publications. Virtualization technology allows additional modularization of applications, which makes services easier to maintain. The bottom line result of using virtualization technology in the server room is increased reliability and better utilization of high end servers, which results in reduced operating costs.[4] This paper, however, considers a completely different application of virtualization technology. It addresses the utilization of virtualization technology in the class room and shares first hand experiences from using virtualization to facilitate student access to the Linux operating system at both the user and super-user (root) security levels. Students were given options to use either traditional methods to access Linux based computers or to use one of two different virtualization technologies to complete assignments in three different classes. In many classroom situations, virtualization offered a preferred solution, but in other situations, a traditional approach was preferred. The objective of this report is present information to assist instructors and systems administration staff in planning for situations where students need to use other operating systems than what is available on laboratory computers.

An informal, *path of least resistance*, experiment was conducted in three classes with distinct assignment objectives. Each class has now been conducted two or more times with students given the same options each time and quite consistent patterns were observed regarding the technology choices made by the students. The primary factor affecting the technology choices was the requirements of the work being performed, which includes the required security level for the assignment, the amount of flexibility required and the duration of the activity. The abilities and preferences of the student was surprisingly a secondary factor with much less impact on the choices made. The students' level of Unix / Linux experience varied quite a bit, but experience tended to only impact their initial attempt, but not their final mode of access. The ease-of-use afforded by a graphical user interface was consistently less important to students than work flow issues. Especially important was the ability to save data so that in future sessions work can be quickly resumed. A few students in this observation pool were non-traditional students with full time jobs. These students were constrained regarding how much time they could spend on campus using the school's laboratory computers. However, the need to use their own computer to complete the assignments while mostly off-campus did influence their technology choices.

OUR COMPUTING ENVIRONMENT

The computing environment at K-State at Salina is quite typical of small under-graduate college settings. The campus has multiple computing laboratories available for student use. The laboratories used for most student course work are equipped with late model desktop computers running up-to-date operating systems from the Microsoft Windows family of operating systems. These lab machines are protected from unauthorized software installation and configuration changes by a product called Deep Freeze.[2] Whenever a user logs out, Deep Freeze deletes files added and restores any files that were changed or deleted by the user.

There is also a networking lab, which is behind a very restrictive firewall. This lab is used by the computer networking and systems administration classes that need administrative privileges. Several of the computers in this lab have removable hard drive systems, which allows students in certain classes to have their own hard drive to use to complete the class assignments. There is also one server grade computer, which runs Linux, available to students in certain classes via remote SSH connection. A few students and faculty have Linux or Apple Macintosh computers for their own use, but most students use Windows exclusively. The servers supporting the campus information technology needs use a mixture of Windows, Linux and Sun Solaris operating systems.

VIRTUALIZAION AND TRADITIONAL TECHNOLOGIES

In simple terms, **virtualization** means the simultaneous execution of more than one operating system (OS) instance on a single computer. This differs from traditional multi-programming, which is just multiple processes executing at the same time. There are several advantages to simultaneous execution of multiple operating systems. Today's computers are so fast that the CPU of most servers are under-utilized if it is running a single OS, even with multiple services running. Yet at the same time, services are

increasing complex in terms of their software configuration, so there is a push to run only one service on each OS instance. With virtualization, multiple, single service OS instances can run from a single computer. It is also possible to migrate these modular OS instances from one computer to another to support maintenance activities.

The available vitalization software products can be grouped into two technology categories. The simpler model uses a host operating system and the virtual OS instances, also called virtual machines (VM), run as an user level processes of the host operating system. The second, newer architecture relies on technology called Hypervisor to eliminate the host operating system. In the later architecture, there is a small piece of operating system like software that controls the immediate access of each virtual machine to the hardware, but it is not a full operating system and uses only a small amount of the available memory and CPU time. With Hypervisor technology, the collective performance of the virtual machines is improved and better features are available to manage the virtual machines. The first, and most well know, Hypervisor product is VMWare's ESX Server.[8]

Hypervisor technology is preferred for network servers run by large organizations. We are interested in teaching students how to work with the products using Hypervisor technology. However, from an education perspective, the older, hosted technology is actually preferred in most cases. This is primarily due to security concerns, but the administrative issues are also simpler when a host operating system is used. When the virtual machines are run using a host operating system, the whole VM has no more privileges than any other user level process. Thus, even if the user is logged in as root (super-user) using a VM, they have no additional capability to engage in malicious activities or to do harm to the computer or the network.

A number of host based virtualization products are available. From the perspective of the user, many of the products provide very similar capabilities. Our criteria for use was that it had to be free and that it had to allow Linux to run as a virtual machine. We used the VMWare Player product to run Linux using Windows as the host computer. We also used User Mode Linux (UML), which is an open source project, to run a Linux VM with Linux as the host operating system. VMWare Player was run using the laboratory computers while the use of UML used a remote SSH connection to a Linux server. As described below, these two products offer opposite ends of the spectrum in terms of ease-of-use and flexibility. Other available software products offer similar capabilities to these two products.

VMWare Player

VMWare Player is a free, desktop application for running a virtual machine.[7][8] The user has full access to the graphical desktop environment of the VM. The virtual disk drive of the VM is a file on the host computer. This file is an image of a drive containing a Linux file system. VMWare's web site contains a number of virtual machine images that be downloaded for immediate use. With the use of the VMWare Server product, which is also free, a custom image can be created. A virtual private network is created between the host computer and the VM. Thus, using the host computer as a router between the virtual network and the computer's network interface, the VM has full

access to the network. Other computer drives, such the CD/DVD player and removable USB drives, are also accessable to the VM. Users have the option of logging in as a regular user or as a super-user.

The primary advantages of VMWare Player for education are that it runs locally on the lab Windows computers and that students have access to the graphical user interface of the Linux VM. Students with limited Unix / Linux experience were attracted to the graphical interface, especially the graphical text editors.

In our environment, we did have the disadvantage of not being able to permanently save data to the disk drive image. The Deep Freeze program restores the disk image after the student logs out. Students are accustomed to saving all of their work on USB based removable storage drives. However, in some cases, students may have to make configuration changes to the Linux environment in order to complete assignments. Thus, assignments that can not be completed in one session required repetition of some configuration changes when the student returned to continue working on an assignment. Printing from within the VM is also not possible. To print a file, the student would need to either transfer the file to the host computer or save it to their removable drive to print later.

User Mode Linux

User mode Linux (UML) is the implementation of the Linux kernel as an user level Linux process.[3][6] Thus, unlike most visualization products, the UML system calls are not trapped by the virtualization software and translated to modified native OS system calls. Because the UML virtual machines run as native Linux processes they have very good performance. However, there are more steps which must be completed before the UML may be used. As with VMWare Player, a disk image of the Linux system must be created. Additionally, a custom Linux kernel must be configured and compiled from the Linux kernel source code. If the UML is run within the Linux desktop environment, it is possible to use the graphical environment of the VM; however, in our environment, which accessed the host computer via a SSH remote connection from the Windows based computers, users were restricted to a command line shell environment. Despite the additional effort required to get started using the UML and the lack of a graphical user interface, UML offers some advantages that make it particularly appealing for many tasks.

Copy on Write (COW) Files

When the UML is started, multiple disk image files may be specified along with the mount points for each. Moreover, for each image file a copy-on-write (COW) file may also be specified. Using COW files, the disk image files can be read-only, with any changes made to the file system being recorded in the COW file. The file system presented to the user, appear to function as any other file system, but any changes made to the file system are recorded only to the COW file. Thus, each student starts with an identical file system, yet the changes made from session to session are re-applied each time UML is started. Students may also start multiple UML instances with each machine

using its own COW file and thus having an unique configuration. If mistakes are made to where it would be easier to start over than to correct previous mistakes, the user can shutdown the UML instance, delete the COW file and start over with a fresh COW file. It is also possible to run commands from the host Linux system to merge a read-only disk image with a COW file. This is a convenient way for system administration staff to test new software or configuration changes. If the changes are acceptable, the file system changes from the COW file may be merged. In an educational setting; however, merging images would not often be needed.

Virtual Network Configuration

In addition to being able to run distinct UML virtual machines with unique configurations, each VM may also use a flexible combination of network interfaces. Rather than starting each VM with a standard, pre-configured virtual network interface, a virtual device called a TUN/TAP, which is effectively a virtual Ethernet cable, is created before the UML is started.[1] In the simplest configuration, one end of the TUN/TAP might be connected to the host system and the other end connected to a VM. Another program, which is a virtual network switch, may also be started. So, a student might build a virtual network using a switch connected to the host system and several virtual machines. One VM might also have multiple network connections and function as a router or firewall between different sub-nets. Thus, the advanced UML user can configure an endless combination of virtual networks and computers to test a configuration in a completely safe, virtual environment.

SSH Client (Putty)

Putty is a simple Windows application that provides a basic console and SSH client for remote connections to Unix / Linux based computers running a sshd server.[5] With *Putty*, students are able to use a Windows based computer to access Linux using the command line interface. One advantage of *Putty* to students on our campus is simply that it is a Windows application which can use the Windows clipboard to facilitate copying and pasting data between Windows applications. How students use this feature is discussed in the description of the *Introduction to Unix* class below.

Individual Workstation Installation

Another option is to install Linux on individual lab computers. In years past, this was done by configuring the boot manager of the computer to provide a menu for selection of booting either Windows or Linux. The advantage of duel booting is that it provides a way to use a basic lab computer for use with either Windows or Linux. The Linux user also has the benefit of the Linux desktop graphical environment. However,

[1]A TUN/TAP is functionally similar to a named loopback interface or named pipe .A data structure is created in the kernel of the host OS and kernel code is used to allow inter-process communication via this device.

the duel boot configuration has been discontinued on our campus in favor of the other technologies presented in this report. The most significant problem with duel booting is that it is impossible to protect the disk partitions belonging to the operating system not currently booted. Also, duel booting causes confusion for the less experienced students and the mechanism for switching between Windows and Linux is slow and cumbersome.

If a Linux desktop environment is needed to complete assignments, a better alternative is to use removable hard drives. These drives use locking trays with all connectors on the back side of the tray, so the user simply slides the drive into the tray, locks it and turns the computer on. For relatively small classes, each student can have their own drive for the semester. This way, the same computer can be used for both Windows and Linux. A compelling advantage of this approach is that the student can build their configuration as the semester progresses. This, of course, implies that the student has administrative privileges on their workstation, which requires that the computer be in a Networking Laboratory, which is behind a restrictive firewall.

CLASSES, ASSIGNMENT REQUIREMENTS AND TECHNOLOGIES USED

Three classes were required to use Linux to complete varied assignments. Here the nature of how Linux was used is described and which access technologies worked best is discussed.

Introduction to Unix Class

This class teaches students how to use Unix / Linux. The objective is to teach new skills, which are needed to work in the information technology field. Thus, the focus is on the command line shell interface. The only real advantage to using a graphical interface for this class is to use a graphical text editor, although students would still be expected to become familiar with a console based text editor.[2] Students only need user-level access to a Unix / Linux computer.

Early in the semester, most students in this class try *VMWare Player* because of the graphical interface, but when they understand that the graphical interface offers few advantages towards completing the assignments, most students find that *Putty* allows the most effective means of completing assignments. The assignments for this class are task oriented. The students are asked to complete a number of activities that are similar to the daily activities of a programmer or a systems administrator and make use of specific standard Unix utilities. Most individual activities can be completed in fifteen minutes or less, but to keep up with the course pace, students must complete three to four activities

[2]Students are introduced to the *vi* editor, which is the one editor that can always be found on a Unix like operating system. However, most students feel more comfortable using an editor where they are always in insert mode, so the simpler, but less functional editor *nano* is the text editor most often used by students. *Nano* is a clone of the *pico* text editor, which is used in the *pine* e-mail client. But because of licensing issues, *pine* and *pico* are not readily available for most Linux distributions, while *nano* is available as a standard package.

by the end of each class period. Students must turn in documentation showing that they have completed each activity. The documentation shows the commands they entered including the content of any shell scripts, which they wrote, and the output produced by the computer. Thus, the simplest work flow seems to be to open the assignment page with a Windows based text editor, such as *Notepad*, and use *Putty* to access the Linux system remotely. Once each activity is successfully completed, the student uses the Windows clipboard to copy from the *putty* console to the text editor to fill in the answer for each activity.

Operating Systems Class

Students in the *Operating Systems* class have used Linux to write, install and use Linux Kernel Modules with the objective of learning more about the internal working and data structures of a modern operating system.[1] As with the assignments for *Introduction to Unix*, a graphical user interface is only a small benefit to the student. However, some students in the *Operating Systems* class have not previously taken *Introduction to Unix*, and thus were quite inexperienced regarding how to use Linux. To install the kernel modules, students need root level access to the Linux system, thus they were not able to simply use *Putty* by itself. For this activity, most students found that *Putty*, in combination with *User Mode Linux* (UML) was the simplest to use. The primary advantages of the *UML* over *VMWare Player* was the ability to save work completed using a COW file versus the requirements of copying all files to removable storage since *Deep Freeze* would restore the disk image to its previous state upon log-out. One variation, which might have made using VMWare Player easier to use for this class would have been to hold the entire disk image on removable storage. However, these disk images can be quite large, so the availability and price of larger (8 GB) USB removable drives, as well as the speed of the USB 2.0 interface, limited the appeal of this option. A few students that were quite uncertain about using Unix / Linux had an initial preference for *VMWare Player*, but when observing the productivity of students using *UML*, most of them eventually switched to *UML*. The exception to this was the non-traditional students with full-time jobs and their own laptop computers. These students took the extra steps of installing *VMWare Player* on their computer. For these students, the driving force of their choice was the need to work on assignments when not on-campus.

Unix Administration Class

This class is a continuation of *Introduction to Unix* and focuses on how to perform Unix / Linux systems administration tasks rather than just using Unix. These students were given the option of installing their own Linux system using a removable hard drive and all students have chosen to do so. Use of the *VMWare Player* in our public labs would not be practical for this class because of *Deep Freeze*. Modern Linux systems include a number of graphical administrative tools that were useful to the students, however, students also learn how to perform administrative tasks by modifying configuration files and issuing shell commands. *User Mode Linux* has also been successfully used in this class for certain projects. *UML* was most useful to this class for

networking related projects, such as building firewall systems.

CONCLUSION

Virtualization technologies used in conjunction with traditional technologies offer appealing options to access Unix / Linux systems as needed to complete course assignments. Having multiple access technologies available is important because the nature of different assignments can significantly impact which technology is preferred. It is also important to note that for certain activities the traditional options may be preferred. Key factors which universally influence the overall ease of use relate more to storing work between sessions than to any convenience offered from a graphical user interface.

BIBLIOGRAPHY

[1] Bower, T., Using Linux Kernel Modules for Operating Systems Class Projects, *Proceeding of the 2006 ASEE Annual Conference (ASEE'06)*, 2006.

[2] Faronics, Deep Freeze Product Page, *http://www.faronics.com/html/deepfreeze.asp*, retrieved July 18, 2009.

[3] Dike, J., *User Mode Linux*, Upper Saddle River, New Jersey, Prentice Hall, 2006.

[4] Virtualization Meets Reality, *Network World*, Special Report, 2007.

[5] Tatham, Simon, Putty Web Page, *http://www.putty.org/*, retrieved July 18, 2009.

[6] UML Project Team, UML Project Home Page, *http://user-mode-linux.sourceforge.net/*, retrieved July 18, 2009.

[7] VMWare Corporation, VMWare Player Product Page, *http://www.vmware.com/products/player/*, retrieved July 18, 2009.

[8] Zimmer, Dennis. *VMware Server and VMware Player*. Sunny Edition, Maur, UK, 2006.

CREATING A SUMMER PROGRAM TO ENGAGE STUDENTS*

Dr. Jerome Eric Luczaj
Department of Computer and Information Technology
Miami University Middletown
4200 East University Boulevard
Middletown, OH 45042
(513) 727-3292
luczajje@muohio.edu

ABSTRACT

The Summer Science Adventures provided an opportunity for rising sophomores and juniors to study discipline specific coursework in an open inclusive environment. This paper will discuss four principles used to create a cohesive curriculum to provide thought provoking, inquiry-based courses to spark student interest, inspire learning, and build confidence.

INTRODUCTION

Over the past nine years, Summer Science Adventures, a summer honor institute, has provided college level courses for gifted rising sophomores and juniors (sophomores and junior in the upcoming school year) high school students. These programs resulted in engaged, knowledgeable, confident students enthusiastic about learning. The lessons learned can be generalized and applied across a wider range of students and activities.

Entering the program, students selected one of five courses offered that summer. Some of courses offered during the program were: Hollywood Mythbusters (a course that used computer technology to explore physics in movies), Computer Animation: Modeling and Motion, Flash and ActionScript Programming to Create Interactive Web Pages, Programming Virtual Worlds (a course that taught object-oriented programming using a three-dimensional virtual world), Hands on Design: Creating Architecture, Crime Scene Analysis through Forensic Science, and Digital Fiction (a course that blended digital technology with creative writing and pictorial story-telling). Every course, whether it was a computer science course or science or art course, relied heavily upon computer technology to deliver course content. The courses were multi-week, full day classes

where students experienced in-depth learning of college level material. Although students received feedback on their work, students did not take tests nor were they required to do any homework.

METHODOLOGY

The goal was to provide thought provoking, inquiry-based curriculum that would spark student interest and enthusiasm, inspire ownership in student learning, and build student confidence through success and a support system of peers.

To accomplish this goal the program focused on four principles: Academically challenging material, Active, hands-on learning, Student interaction and collaboration, and the Opportunity to share their work. This paper will address each of these areas in turn.

As discussed by Becker [1] applying Merril's first principles of instruction will greatly enhance the effectiveness of computer science instruction. These include engaging in real-world problems, using existing knowledge as the foundation, demonstrating new knowledge, and applying new knowledge, integrating it into the student knowledge base. In other words, challenge the students with new material that connects to their own by giving them examples that they can show to others. In addition, many studies have demonstrated that active learning is an effective means to learn [2][3] as is student collaboration [4][5][6].

Academic Challenge

The material should provide an academic challenge to the students. To fully engage students in the program, the content should stretch the students into areas they have not yet explored. The belief is that programs, especially where students give up a large portion of their summer break, spending their free time or extra-curricular time, need to give the students experiences and knowledge unavailable to them during their normal school year.

However, it is equally important that students have successes all the way through the program. This was accomplished in multiple ways. First, material was broken into small units each with tangible results. The overall course structure grouped these smaller units into a larger, connected experience that provided the knowledge and skills that the students needed to create a comprehensive final project. For example, in the Programming Virtual Worlds class, students learned about variables and dynamic object instantiation by creating a puzzle that, when solved, gave their character a key to open a locked door. Then they built upon this by using loops to create a fixed number of penguins in the world when the door was opened.

Second, material was accessible. Courses were taught in a student-centered manner. Examples related to the students. Students drove project selection choices. Further, students were guided within the framework of the course to make independent choices and take responsibility for their learning. For example, to decide a topic for their final project in the ActionScript course, the students brainstormed on projects they would like to do. The instructor helped them scope and rank the projects by level of difficulty.

Students selected project teams and projects from this student developed, instructor-guided list. As a result, the students took on difficult projects that stretched their abilities, but with a sense of ownership and enthusiasm.

By providing new, challenging, academic content delivered in a structured, student-driven, accessible manner the program linked success to academic challenge and built student confidence. Students responded extremely favorably. Typical comments were: "The content in this course was much harder than high school, but (the content) was a lot more interesting and fun." and "…I was treated like an adult and not like someone that is not capable of thinking for themselves…"

Both of the examples given here were specific to courses in computer science, but can be applied to other disciplines. Students will rise to academic challenge with enthusiasm especially when the instructor connects the challenge to student interests and helps break the material into manageable units. Success without challenge becomes tiresome, success with challenge is energizing.

Active, Hands-on learning

Student engagement occurs when students are involved in the class session, when they are involved in class activities, attentive and participating. It only follows, then, when students perform hands-on activities that they are engaged with the material. All courses within the program relied heavily upon hands-on exercises and projects to motivate student learning. Each exercise resulted in some tangible artifact, created by the students. For example, the first mini-project in the Animation course resulted in an animated, 3-D logo designed and created by the student. Again, these hands-on activities relied upon a student-centered model to foster pride of ownership and motivate student enthusiasm. It was not uncommon for students to try to work through their morning and afternoon breaks because they were so absorbed in their projects (and the learning that these projects entailed).

As previously mentioned, the program also actively involved students in course and project decisions. As seen in studies (like Radenski [7]) giving students choice also increased their motivation to perform. However, this was not the only benefit of working with students on class decisions. It also gave students a model to use and practice developing planning and decision making skills to help them in all their future activities. For example, after deciding as a class to create a published anthology as the final project for the Digital Fiction course, the class determined tasks needed, grouped them into milestones, set responsible parties, and set the schedule with deadlines. The result was hard-working, actively engaged students that created a 193 page, self-published graphic novel anthology.

As in these examples, in computer science courses, involving the students in both the activities and the decision making process will result in a more motivated student because they are both responsible for the creating the final product as well as determining how the product will be created.

Student Interaction and collaboration

People, especially at the targeted age group, tend to be very social. The program tapped into these tendencies with ice-breaking activities, small group projects, group shuffling, and social time.

Ice-breaking activities were designed to start building a community in the classroom. When possible these were tied to the course content, but the over-riding goal was always for the students to get to know one another. One example of tying an ice-breaking activity with course content was accomplished in the Digital Fiction class. Students were paired and told to interview each other about what they wanted to do in the future, but not about what they had done in the past. Then, using the gathered interview material, students wrote stories about a character that had succeeded with those future goals but with a fictional past. Students read the results out loud to the class. As a result, these introductions were more engaging. Initial contact with each other was in small groups and the interview pair helped provide support for reading their work out loud to the class. Though there was added benefit received, the intent was for students to establish a comfort level with each other.

The program also was designed so that students had a mixture of group and solo projects, tending to mostly small group projects, with a target group size of two to three students. The small group model helped the students establish a rapport and required that each student work to get the tasks accomplished. As mentioned above, most of these projects were small scale learning projects designed to bring success to the teams. As a result, not only did students get to know each other better, but also they learned that working with others produced success. It was not enough, however, to learn that working with a particular teammate brought success, students must be able to extend that good experience to working in teams in general. This student's comment was typical of student reaction to working in teams "…not only did we learn from other people, we learned how to work together with them even though we had only known each other for a short amount of time."

The intent was to build connections between all of the students. Consequently, the small groups were shuffled for every project. Only after they have had the opportunity to work with all the other students at least once were students given the chance to choose their own teammates. When they did get to choose teams, their choice was an outward sign of their connection to others in the class and created a deeper, lasting connection to the people in the program.

Lastly, the program set aside social time for the students outside of the academic setting. Time for social interaction included breaks in the morning and afternoon session as well as an extended lunch period. Board games, card games, disc golf, and soccer balls were all provided for the students to use. There were also a limited number of areas for the students to go, which did not include an area with a television. The goal was active student interaction, not passive co-location. The other advantage with the social time was that it helped to break up the day and helped the students maintain focus while they were in the classroom.

Especially in computer science, students will be expected to work together throughout their careers. Large parts of teamwork are the trust relationships that the team

members develop. Activities, like those described above, can be applied to any course to help motivate and engage students.

Opportunity to share their work

The last area that the program used to motivate students was the opportunity to share their work with others in small and large forums. Time was consciously scheduled throughout the program during the class sessions for students to share their in-progress and completed work. These opportunities tended to begin with informal sharing such as reading the fictional biographies in the ice-breaker activity mentioned above and progressed to more formal presentations of their work in front of the class. Easing the students into formal presentation as their confidence and comfort progresses was another way to give practical, practiced life-skills to students in the program.

The keystone of the program, though, was the final day presentation. Students invited family and friends to see them present their final projects, making these presentations a celebration of their experiences and accomplishments in the program. This did at least four things. First, it motivated the students to apply themselves in a very directed, productive way. Second, it gave students practical experience presenting their ideas to a large audience. Third, it provided a final transition from course content that belongs to the program to content that was created, internalized, and owned by the students. Fourth, it connected the student's feelings of success, in a very tangible way, to the program.

Bringing this idea into a computer science curriculum would allow the students the opportunity to show pride in their work and accomplishment. This would require that instructors insure that every student had a part in some tangible, demonstrable, concrete product. That each student had taken ownership of what they and their team had created.

RESULTS

So the question remains, did this approach work? Specifically, did this approach engage students in learning? And did they learn? During 2007 and 2008, the Ohio Department of Education administered anonymous online surveys of students in the program. The survey asked students to rate various aspects of their experience in the program using the 5-point scale: Strongly Agree, Agree, Disagree, Strongly Disagree, Not Applicable (were Strongly Agree was the most positive response and Strongly Disagree was the most negative response). Of 610 total student responses 398 (65%) answered Strongly Agree, 211 (35%) answered Agree with 1 response of Not Applicable. Student comments reinforced this extremely positive impression. Examples included "I loved it! I'm not the only nerd out there! I made friends with students I wouldn't otherwise have the chance to meet." and "…the course truly allows one to walk away with something one can make real use of sometime in the near future."

Parent and teacher surveys were likewise positive. Parent comments included: "Gave my son something very positive, challenging, and educational to do during the summer…" and "wonderful opportunity; my daughter was incredibly enthusiastic about each day's learning experiences." Teacher comments included: "My student was smiling

the entire time. I think he enjoyed being with other kids just as interested as himself…", "…I was impressed how well he presented. I think this added to his confidence.", "It challenged them, it made them use their knowledge in a creative way, and gave them a sense of accomplishment." and "My student was highly encouraged about her future. She seemed far more interested in science."

To answer the question: Did the students learn, pre and post assessment evaluated student understanding of key content concepts within each course. Evaluations were on a 4-point scale: Haven't heard of these terms, Heard of terms but can't explain, Can explain but not demonstrate, Can explain and demonstrate. During the 2007 and 2008 programs the average pre course score was approximately 1.4 (ranging from 1.2 to 2.3 depending upon the course), indicating that the students had mostly not heard of the terms before and if they had, they could not explain them. The post course score was approximately 3.5 (ranging from 3.2 to 3.9 depending upon the course), indicating that students could at least explain the terms and most likely demonstrate the concepts within the context of the discipline.

Applying the principles of Academically challenging material, Active, hands-on learning, Student interaction and collaboration, and the Opportunity to share their work, has shown to be effective in student learning as well as with student reaction and attitudes.

As one student put it, "I had hands-on lab experience in this course that I will never forget. I made friends and had a lot of fun."

REFERENCES

[1] Becker, K. 2006. First principles of CS instruction. *J. Comput. Small Coll*. 22, 2 (Dec. 2006), 77-84.

[2] McConnell, J. J. 1996. Active learning and its use in computer science. In *Proceedings of the 1st Conference on integrating Technology into Computer Science Education* (Barcelona, Spain, June 02 - 06, 1996). ITiCSE '96. ACM, New York, NY, 52-54.

[3] Schweitzer, D. and Brown, W. 2007. Interactive visualization for the active learning classroom. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (Covington, Kentucky, USA, March 07 - 11, 2007). SIGCSE '07. ACM, New York, NY, 208-212.

[4] Gonzalez, G. 2006. A systematic approach to active and cooperative learning in CS1 and its effects on CS2. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA, March 03 - 05, 2006). SIGCSE '06. ACM, New York, NY, 133-137.

[5] Ludi, S., Natarajan, S., and Reichlmayr, T. 2005. An introductory software engineering course that facilitates active learning. In *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA, February 23 - 27, 2005). SIGCSE '05. ACM, New York, NY, 302-306.

[6] McKinney, D. and Denton, L. F. 2006. Developing collaborative skills early in the CS curriculum in a laboratory environment. *SIGCSE Bull*. 38, 1 (Mar. 2006), 138-142.

[7] Radenski, A. 2009. Freedom of choice as motivational factor for active learning. In *Proceedings of the 14th Annual ACM SIGCSE Conference on innovation and Technology in Computer Science Education* (Paris, France, July 06 - 09, 2009). ITiCSE '09. ACM, New York, NY, 21-25.

TEACHING MATHEMATICAL PROOFS TO CS MAJOR STUDENTS IN THE CLASS OF DISCRETE MATHEMATICS *

Hongbiao Zeng, Department of Mathematics & Computer Science,
Fort Hays State University, Hays, KS 67601 (785) 628-5811 hzeng@fhsu.edu
Keyu Jiang, Department of Informatics,
Fort Hays State University, Hays, KS 67601 (785) 628-4684 kjiang@fhsu.edu

ABSTRACT

Teaching mathematical proofs to computer science major students is a challenge. Computer science majors like logical reasoning but not complicated mathematical proofs. This paper proposes two possible effective ways to teach mathematical proofs to computer science major students in the class of discrete mathematics.

INTRODUCTION

The fundamental concepts of modern computer science are built almost entirely upon discrete mathematics. In order to learn the fundamental algorithms in computer science, computer science students must have a solid background in discrete mathematics. Indeed most universities and colleges have listed the course of discrete mathematics as a requirement for pursuing a Computer Science degree. Of course, the class of discrete mathematics may have various names in different universities and colleges. In some universities and colleges, it is called Foundations of Computing. The name itself explains how important discrete mathematics is to computer science.

Like other courses in mathematics, there are many mathematical proofs in discrete mathematics, even when the course is particularly designed for computer science majors. It is important to understand those proofs in order to understand the related theorems, formulas, and concepts. However, since the audience is computer science majors, who are curious about the logical reasoning but not too comfortable with the complicated mathematic proofs, teaching these proofs to those students is always a challenge. This

---

paper researches some ideas on dealing with this challenge.

This research originated during one of Zeng's classes while the recurrence relation was being taught. When Zeng was showing how to find the solution of a recurrence relation, a famous identity was cited:

$$1 + 2 + 2^2 + \cdots + 2^n = 2^{n+1} - 1$$

One student suddenly raised his hand and asked why the identity is true. This easy question is not easy to answer. On one hand, a simple answer such as "it is a formula" will not satisfy the students who want to have a logical reason. On the other hand, a proof using mathematical induction (or other calculation) will just cost too much time and distract the students from the original purpose of that class. Fortunately, Zeng came up with a solution which satisfied the computer science students immediately. Consider the ***binary*** number system, $(1 + 2 + 2^2 + \cdots + 2^n)_{10}$

$$= \underbrace{11 \ldots 1}_{n+1\ 1s}{}_2$$

$$= \underbrace{100 \ldots 0}_{n+1\ 0s}{}_2 - 1$$

$$= (2^{n+1} - 1)_{10}$$

In less than one minute, the question was answered and the class went back to its original path.

This good experience motivated us to think about the following question: how do we teach mathematical proofs in a way that attracts computer science students and is easy for them to understand? In this paper, we propose two such ways, namely, teaching with CS thoughts and teaching with graphs.

TEACHING MATHEMATICAL PROOFS WITH CS THOUGHTS

What is a CS thought? We define that a CS thought is any computer science related thought with which computer science students are familiar and comfortable. If a mathematical theorem or formula can be proved using a CS thought, it will be easy for computer science students to understand and accept.

We believe that there are many CS thoughts out there that can be used to prove mathematical theorems and formulas. In this paper, we will discuss three CS thoughts that we have used in our Foundations of Computing class. These three CS thoughts are thinking of binary, using truth tables, and writing algorithms.

Thinking Of Binary

Thinking of binary is an obvious CS thought. As it is said in a famous computer science joke: there are 10 kinds of people in this world: the one who understands binary and the one who doesn't. Computer science majors belong to the first kind who understands binary. In fact, computer science majors are familiar and comfortable with the binary number system. This explains why the proof cited in the introduction section was accepted immediately by students.

$$\frac{1}{2}+\frac{1}{2^2}+\cdots+\frac{1}{2^n}=1-\frac{1}{2^n}$$

It is easy to find another similar example: using binary thought to prove the identity

$$1+r+r^2+\cdots+r^n=\frac{r^{n+1}-1}{r-1} \qquad (1)$$

Are there any more examples? Let's consider the following identity

where $r \neq 1$.

When we explained the above identity to our students, we pointed out that thinking of binary implies that the base doesn't have to be ten. If r is an integer that is greater than 1, we can consider the number system of base *r*. Hence

$1+r+r^2+\cdots+r^n$

$$=\underbrace{11..1}_{n+1\ 1s}{}_r$$

$$=\frac{\underbrace{100...0}_{n+1\ 0s}{}_r-1}{r-1}$$

$$=\frac{r^{n+1}-1}{r-1}$$

Since *r* could be any real number except 1, the above argument is not a complete proof of identity (1). However, it is enough to convince a computer science student and make the identity memorable.

Using Truth Tables

Using truth tables instead of the logical arguments is also a CS thought. Many authors have used it to prove some of the Boolean formulas in their textbooks of discrete mathematics ([1], [4]). However, we still can dig even further if we want. For example, the proof of the Generalized De Morgan Law for Logic in [1] is completed by using logical argument. We proposed the following proof in our Foundations of Computing course:

The Generalized De Morgan Law for Logic: $\overline{\forall x, P(x)} \Leftrightarrow \exists x, \overline{P(x)}$

Proof: See truth table, where $\forall x, P(x)$ acts as a bridge.

$\overline{\forall x, P(x)}$	$\forall x, P(x)$	$\exists x, \overline{P(x)}$
True	False	True
False	True	False

Although the table will be interpreted exactly as the logical argument presented in [1], the students felt more comfortable with the table simply because they are familiar and comfortable with truth tables!

Writing Algorithms

Writing algorithms is a typical CS thought. Being able to write algorithms to solve problems is a fundamental requirement for computer science majors. Asking the students to think of an algorithm to solve a problem is always a good practice in teaching computer science majors. When we were teaching the section of mathematical induction in our Foundations of Computing course, we used the following example to show how mathematical induction works:

Show that $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ for all positive integer *n*

We first asked the students if they could write a computer program to prove this identity. After correctly receiving a "NO" answer, we asked the students if they could write a computer program to prove the identity up to a given N (not too big). This time, the students answered "yes". Then we asked for the algorithm. An algorithm was proposed immediately:

write a computer program to prove the identity up to a given *N* (not too big). This time, the students answered “yes”. Then we asked for the algorithm. An algorithm was proposed immediately:

Input: N

Output: true if $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ for all positive integers upon to N; false otherwise

```
Procedure checkIdentity(N)
sum = 0
for n from 1 to N
        sum = sum + n
    if sum ≠ n(n+1)/2 return false endif
endfor
return true
```

After applauding the success of writing the algorithm, we asked the students to observe that the *(k+1)*th iteration of the for loop depends on the *k*-th iteration of the for loop. If we can show that the correctness of *k*-th iteration of the loop will imply the correctness of *(k+1)*th iteration of the loop for any *k*, then the algorithm will return true for any positive integer N since the loop's first iteration will always pass without returning false. This is exactly the idea behind mathematical induction!

TEACHING MATHEMATICAL PROOFS AS GRAPHS

Graphs are always good tools in human communication. In computer science, graphics are especially important. For instance, a set of graphic diagrams can depict the software to be built from its requirements to its design. We can safely say that if the proof of a mathematical theorem or formula can be shown as a graph, it will be easily accepted by a computer science major student. The proof shown as a graph is called "proof without words."

"Proofs without words" are often beautiful and intuitive. To share these proofs is a great enjoyment among mathematicians. In fact, two major journals published by the Mathematical Association of America, namely *Mathematics Magazine* and *The College Mathematics Journal*, have published hundreds of such proofs under the category named "Proof without Words." Examples of "Proof without Words" can be found in [2], [3].

Examples of "Proof without Words" can also be found in some textbooks of discrete mathematics, although they are not referred to as "Proof without Words." For instance, the Venn diagram is used to prove some identities related to sets in [5]. The Venn diagram is also used to represent the concept of $\overline{A \cup B}$ in [6], which implies the proof of one of De Morgan's Laws for Sets: $\overline{A \cup B} = \bar{A} \cap \bar{B}$.

We had a good experience in "Proof without Words" when the topic of complexity of algorithms was taught. The students were very confused with big O, big Ω and big Θ notations. We used the following example to explain these concepts:

$$1 + 2 + 3 + \cdots + n = \Theta(n^2)$$

In order to quickly establish the inequalities satisfying the definition of big Θ, we simply showed the following graph instead of the proof in [1].

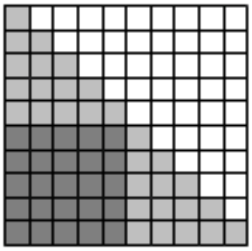


Figure 1

Although the graph is just for $n = 10$, it illustrates the general idea for any positive integer . From the graph, it is easy to conclude that the area of the dark part is less than the area under the diagonal (including the diagonal), which is less than the area of the biggest square. This conclusion is equivalent to $\frac{n^2}{4} \leq 1 + 2 + \cdots + n \leq n^2$. The students immediately concluded that

$$1 + 2 + \cdots + n = \Omega(n^2) \text{ and } 1 + 2 + \cdots + n = O(n^2)$$

which implies $1 + 2 + \cdots + n = \Theta(n^2)$

As we see in the previous example, "Proof without Words" is not a rigorous mathematical proof of a theorem or formula. However, it gives a very intuitive view to understand why the theorem or formula is correct. Here we discuss one more example from combinatory theory in Discrete Mathematics:

$$C_n^k = C_{n-1}^{k-1} + C_{n-1}^k \qquad (2)$$

To prove this identity, we simply draw the following graph:

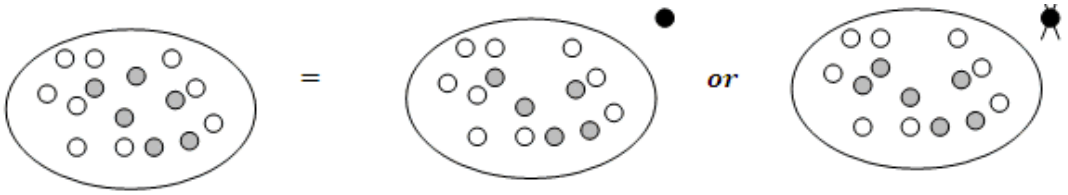


Figure 2

The graph clearly shows that choosing k dots (gray dots) from n dots can be done in two ways, assuming that one dot falls out:

1. Taking the fallen-out dot (black dot) then choose *k*-1 dots from remaining *n*-1 dots.
2. Or not taking the fallen-out (black dot) but choose *k* dots from the remaining *n*-1 dots.

It is exactly the meaning of the identity (2).

CONCLUSION AND ACKNOWLEGEMENT

It requires a significant commitment to teach any course efficiently. As computer science faculty members who had been teaching discrete mathematics to computer science majors for several years, we cannot agree more that teaching mathematical proofs effectively to computer science students requires an even greater commitment. This paper is just beginning research in this subject. Further research should be done with two areas. First, find more theorems that can be proved by using the proposed two methods in this paper. This includes finding more CS thoughts and "Proof without Words" to prove more theorems and formulas in discrete mathematics. Second, find other efficient methods to teach mathematical proofs to computer science major students.

We appreciate the many contributions of our students who enrolled in our course "Foundations of Computing" and provided helpful feedback. We also appreciate our colleagues Ms. Mary Kay Schippers and Dr. Carl Singleton who read through this paper and gave many valuable suggestions.

REFERENCES

1. Johnsonbaugh, R., *Discrete Mathematics*, Upper Saddle River, NJ: Prentice Hall, 2001.
2. Unal, H., Proof Without Words: Sum of an Infinite Series, *The College Mathematics Journal*, Page 39, Vol. 40, No. 1. January 2009
3. Plaza, A., Proof Without Words: Bernoulli's Inequality, *Mathematics Magazine*, Page 62, Vol. 82, No. 1. February 2009
4. Goodaire, E and Parmenter, M, *Discrete Mathematics with Graph Theory*, Upper Saddle River, NJ: Prentice Hall, 2006
5. Zehna, P and Johnson, R., *Elements of Set Theory*, Boston, MA: Allyn and Bacon, 1972
6. Hein, J., *Discrete Mathematics*, Sudbury, MA: Jones and Bartlett Publication, 2003

PVIF OF $1 TABLE CREATION & USAGE IN C# :

FUNDAMENTAL INTRICACIES*

NIFTY ASSIGNMENT

Jean Hendrix, Associate Professor
Computer Information Systems Division
University of Arkansas at Monticello
Monticello, AR 71656

The following project is submitted as a "Nifty Course Assignment" for several reasons. In our Computer Information Systems program students must take a core object-oriented class in which they use Visual BASIC but can only gain experience with C# if they choose to take it as an elective. For those students who do elect to enroll in C# there are many advantages which include a comfort level writing code based upon their experiences in VB and the familiarity with Visual Studio and the creation of event driven systems which proceed from it. Also, by the time students take this course they are generally much more "seasoned" in regard to programming logic and design, and can readily work with problems of a more interesting nature. Unlike computer science majors our CIS students are required to take supportive classes in area such as business and accounting, and when they are equipped with this background it is quite appropriate to formulate assignments which dovetail programming with financial topics.

All accounting students are taught the concept of a Present Value and how it basically represents the discounted value of a future cash flow. In other words it performs conversely to compounding in that it discounts future expected amounts back to a "present day" dollar amount. To calculate a Present Value three data must be known: the future amount to be discounted, the discount rate that applies, and the time frame. To simplify calculations, accounting textbooks include PVIF of $1 tables as appendices and the users only need multiply the correct factor by the future expected value to obtain the full discounted value.

Armed with this information my students are asked first to create a C# program which will calculate the "full discounted amount" based upon the requisite input of 1) future amount 2) discount rate and 3) time frame. This is where the students gain a deeper appreciation of the differences in C# and other languages like Visual BASIC. It is a revelation to them when they discover that there is no arithmetic operator for exponentiation! Even though they have had experiences using and creating methods (and

functions in VB), having to employ the Math.Pow method is a minor challenge. I always try to emphasize mathematical equivalencies so ask them to write the formula in different ways for "show and tell". If the sign on the expression in the denominator for (1/((1+r)^n)) is changed, the entry is equivalent but looks a little nicer. Oh, and by the way, they must be especially sensitive to the "double" data type since the Math.Pow method requires it.

Now comes the fun program: to create another application that generates a PVIF of $1 table internally and then allows a user to select a desired discount rate from a list box control and select a time frame from list box control and input/type the future amount before clicking a command button to calculate the discounted amount.

The first part of this program requires the declaration of a two-dimensional array which syntactically differs somewhat from languages with which they are most familiar. Next a nested "for" statement must be written that will correctly set the subscripts for the table but also, when modified, serve as variables for the discount rate and the number of years. Because C#'s arrays are zero-based the student must be keenly aware of how to set the initial value for the looping variable and how to modify this amount for use in the formula. When initially working with tables one expects to receive an "Subscript Out of Range" error message a few times, too, which is instructive for students to think through. In C# it is especially critical to set the blocks (scope) carefully and this exercise demonstrates it well. Once the "reference table" has been created the second part of the program allows the user to request that the program calculate a Present Value based upon a factor found in the table. The look-up process is straightforward but the student must be aware that only percentage rates (usually the columns) and time frame in years (usually the rows) on the table are viable and must match with the values in their list box controls. For a practical application, of course, this is very limiting but the point of the exercise is to give the student more opportunities to work with multi-dimensional arrays and experience the intricacies of C# syntax. Of course when it serves to reinforce a financial concept, too, what more could we ask!

ROULETTE SIMULATIONS AND MARTINGALE BETTING*

NIFTY ASSIGNMENT

David Reed
Department of Computer Science
Creighton University
davereed@creighton.edu

SUMMARY

Students modify and implement interacting classes in order to simulate repeated roulette games, and use their simulation to study the effectiveness of different betting strategies.

TOPICS

The context for these assignments is class design and interaction, and the use of computer models to simulate real-world events. The program utilizes conditionals, loops, random numbers, strings, counters, and sums.

AUDIENCE

This assignment is given in the second half of a CS1 course using Java, after students have had some experience with class use and implementation.

STRENGTHS

This assignment involves working with interacting classes, as students must use a provided class and also implement a class that is used by another. This forces them to think abstractly and also program to exact specifications. The problem-solving aspect of this assignment emphasizes that most programs are not end-products but tools that are used to solve problems. Using their program to simulate repeated games, test different betting strategies, and analyze their performance, is both motivating and entertaining.

WEAKNESSES

While the student is only required to implement one class and make minor modifications to a method in another class, there is a considerable amount of code that comprises the project and that they must comprehend. Weaker students may initially be overwhelmed and need assistance in reviewing the provided code framework.

ABSTRACT

The Martingale betting strategy for playing roulette is centuries old, but still pops up in viral emails and in various scams. The strategy calls for the gambler to double the bet amount after each loss, so that the first win would recover all previous losses plus win a profit equal to the original bet. The claim is that this system will guarantee a profit. In reality, it does nothing of the kind, since its success relies upon the gambler having an infinite bankroll (and the house not imposing betting limits).

In part 1 of this assignment, the students are provided with the RouletteWheel class that models a roulette wheel and asked to implement the RouletteGame class for playing a game of roulette. A player in the game can enter credits into his or her account and can make bets on spins of the wheel. The player may bet on a specific number (1-36) or a color ("red" or "black"), and the game keeps track of their winnings and losses.

In part 2, the students use their roulette game code to study different betting strategies. They are presented with the question: If you had to double your money (or else), what would be your best betting strategy? By simulating thousands of games using the RouletteTester class, with each game continuing until the player has doubled his or her money or else gone broke, student are able to determine which strategy (number vs. color, big bet vs. small bet) is preferable. The results can be somewhat surprising.

In part 3 of the assignment, students are asked to extend their analysis to the Martingale betting strategy. This involves relatively minor modifications to the simulation code, so that the bet amount changes after each spin. Using the modified simulation, students are able to compare the performance of the Martingale system with previous betting strategies and observe just how ineffective this "can't lose" system truly is.

A full version of the assignment, along with source code, can be found online at `http://dave-reed.com/NiftyRoulette.`

PROVIDING A DIGITAL LOGIC LAB EXPERIENCE IN A COMPUTER ARCHITECTURE COURSE*

NIFTY ASSIGNMENT

James Feher
Division of Computing
McKendree University
Lebanon, Illinois 62254
jdfeher@mckendree.edu

ABSTRACT

The laboratory exercises discussed in this presentation are intended to supplement a traditional computer architecture course. Digital logic labs provide additional opportunities for students with varying learning styles to master Boolean algebra as well as experiment with the building blocks required for any digital computer. However, the cost and procurement of the necessary equipment combined with the fact that faculty may be reluctant to require an extra text that may be necessary to facilitate these labs can act as obstacles in offering these types of laboratory exercises. The author describes how he addressed these concerns by incorporating digital logic labs within a computer architecture course using inexpensive commodity components along with a supplementary text that was released under the Creative Commons Attribution License.

INTRODUCTION

This paper discusses the implementation of digital logic laboratory exercises within the context of a traditional computer architecture course. At our institution, the computer architecture course is taught using a traditional text for the subject [1] with no specified laboratory hours and is composed of students from both the computer science and information systems majors. Providing alternate learning opportunities for the students can often facilitate greater retention and comprehension of the material. And while "students must develop an understanding of the scientific method and experience this mode of inquiry in courses that provide some exposure to laboratory work" [2, p. 41], it is often difficult to provide this experience for a variety of reasons including the expense

of an additional text and necessary equipment. In addition, the course at our institution does not have any laboratory hours specified, so the labs added would need to be completed on an informal basis.

"I hear and I forget. I see and I remember. I do and I understand." - Confucius

In order to address the concerns regarding obtaining a text, the author of this paper wrote a text specifically for this subject [3] that is available under the Creative Commons Attribution License [4]. In writing the text, care was taken to specify the use of nothing more than a standard digital logic kit and inexpensive commodity components that are readily available.

THE TEXT

The text is a lab manual that includes theoretical background, review problems and suggested laboratory exercises. It was developed specifically for the purpose of adding lab exercises in digital logic as a supplement to a traditional computer architecture course. Development of the text was coordinated with the Global Text Project [5] whose aim is to "create open content electronic textbooks that will be freely available". As the text is available under the Creative Commons, it can be freely downloaded as a pdf.

The manual starts with the construction of a simple inverter using a transistor and concludes with the construction of simple state machines. The theoretical content provides the background necessary to introduce each of the concepts, followed by review problems with detailed solutions. Suggested laboratory exercises conclude each section. While the laboratory exercises can be included as part of a formal lab experience, they are designed so that students with a minimal orientation can complete them independently in a modest amount of time. This assumes that the student has read the theoretical background, received a short lecture on the material, completed the review exercises and developed the necessary pre-laboratory design. It was important that students could work independently on the labs as our institution does not have a formal lab that accompanies our computer architecture course.

RESOURCES REQUIRED

Materials and resources were selected based upon their availability and modest cost. First, the various capacitors, resistors and wires required are easily obtained and inexpensive. The integrated circuits required are the 555 timer and 7400 series chips which have been available and used extensively in industry since the 1970s. The 555 can be used to build a low frequency clock for asynchronous circuits without the use of a crystal or ceramic resonator. The 7400 series chips provide the logical gates as well as more advanced circuits such as the multiplexer and quad D flip-flop.

In addition a logic kit consisting of a breadboard, five volt power supply, input switches and leds are all needed to complete any of the designs. While this kit can be built from the parts specified, it is often more convenient to purchase a digital logic kit that is already assembled [6].

SUMMARY OF THE LABS INCLUDED

The Transistor and Inverter

This lab provides an introduction to the use of a breadboard and digital logic kit by constructing a single inverter using a couple of resistors and a transistor. The student then is introduced to the 7400 series of TTL logic chips by using a single inverter from a 7404 chip.

Logic gates

Students are introduced to simple logic gates: AND, OR, NAND, NOR and XOR. Truth tables are used to represent more complex Boolean expressions. Students then build simple logical Boolean expressions using the logic kit and 7400 series chips.

Logic simplification

More complex logical expressions are simplified using Karnaugh maps to yield Sum of Product (SOP) expressions. DeMorgan's laws are discussed as they pertain to implementing SOP expressions with two levels of NAND gates. Rudimentary circuit debugging techniques are suggested.

More logic simplification

Non-obvious Karnaugh map groupings and don't care conditions are discussed. Additional Boolean expressions are implemented using 7400 series chips.

Multiplexer

Logical function implementation using the multiplexer is covered. Three and four input Boolean expressions are implemented using the 8x1, 74151 multiplexer.

Timers and clocks

Frequency and period are discussed. Timing diagrams are introduced. The 555 timer is used to build both a timer and a clock circuit.

Memory

Static memory is introduced in the form of the SR latch. D, JK and T flip-flops are briefly discussed. An SR latch is built with NAND gates and the 74175 quad D flip-flop is used to obtain one bit of memory.

State machines

Boolean logic, clocks and memory are combined to build a simple state machine. State transition diagrams are introduced. Timing diagrams and debounced switches are also discussed. Simple two bit counters are designed and then built.

More state machines

More complex synchronous circuits are designed and built. State machines with an number of states that are not a power of two are designed. Strategies for design simplification and reliability of the final circuit are discussed as they relate to the unused states for the state machines.

METHODOLOGY

The laboratory manual was used successfully during the fall 2009 semester in a three credit hour computer architecture course with no formal lab. Students in the course were drawn from a wide variety of majors from within the Division of Computing at McKendree: Computer Science, Computer Information Systems, Information Technology, Computational Science and Interactive Media. This group has a diverse foundation in mathematics and science, and those with a stronger foundation in these areas master the material more quickly. Prior to starting the labs brief introductions were provided for a few essential elements.

- Provided materials (breadboard, chips, resistors, etc.)
- Introduction to DC electronics
- Theory for each lab

This seemed sufficient for the students to then progress independently.

Students worked with a partner, generally with a weaker student paired with a stronger student, and all labs were completed outside of the class with assistance available to those who experienced difficulty. Before starting to construct a lab, pre-laboratory work in a lab manual was checked to verify that the circuit was properly designed and documented. The final completed lab was then demonstrated to the instructor for each of the labs. The labs were spread throughout the semester with the lectures interlaced with the material from the traditional architecture text and with plenty of time for the students to construct their circuits.

RESULTS

I have been incorporating labs of this nature every time I have taught a course in Computer Architecture. This material has been well received, with student comments generally following that, they enjoyed *"the hands on activity with the labs, it was nice to see the small parts work."* Developing the text has allowed me to include these labs with minimal expense on the part of the students. With regards to this text, comments followed this general theme, *"The lab book was straight forward and easy to comprehend. I enjoyed working and learning from it."* I have found that after completing the labs, even the weaker students in the course demonstrated an excellent grasp of the theory. Some students have commented that they disliked spreading the material throughout the semester and would prefer not to *"switch between two books."* However, I have found that by placing these labs during transitions between chapters this can be minimized. I have especially found that placing the labs during the middle of a particularly difficult concept, such as cache memory design, allows the students have more time to reflect upon and master the more difficult concepts.

CONCLUSION

Despite the fact that very few students had any previous exposure to using a breadboard or digital logic kit, the lab groups were able to successfully master the use of these prototyping tools fairly quickly. Anecdotally, students generally expressed

satisfaction in building the circuits. I found that the students also were able to demonstrate a level of mastery with the material covered in the labs that was far greater than the material from the traditional text when it only was reinforced through pen and paper homework problems. Some suggestions for the text arose, such as providing a few more advanced combinatorial logic circuits such as an adder, before introducing sequential logic circuits. These suggestions will be incorporated in a newer edition of the text. Overall, the inclusion of the labs was a success and I will continue to incorporate digital logic design in this format whenever teaching a computer architecture course.

REFERENCES

[1] Stallings, W., *Computer Organization and Architecture*, Prentice Hall, 2003.

[2] Chang, C., et al, *Computing Curricula 2001- Computer Science*, The Joint Task Force on Computing Curricula, IEEE Computer Society and the Association for Computing Machinery, 2001.

[3] Feher, J., *Digital Logic with Laboratory Exercises*, 2009, http://docs.globaltext.terry.uga.edu:8095/anonymous/webdav/Digital%20Logic/Digital%20Logic.pdf.

[4] Creative Commons – Attributed 3.0 Unported, 2009, http://creativecommons.org/licenses/by/3.0/, Retrieved November 30, 2009.

[5] Welcome to the Global Text Project, 2009, http://globaltext.terry.uga.edu, Retrieved November 30, 2009.

[6] Educational > Logic Trainers - Products (Page 1) - HVW Technologies, 2009, http://www.hvwtech.com/products_list.asp?CatID=206&SubCatID=207, Retrieved November 30, 2009.

INDEX OF AUTHORS

JCSC Volume 25 Number 5 May 2010

Muhlenberg College
2400 Chew Street
Allentown, PA 18104-5586