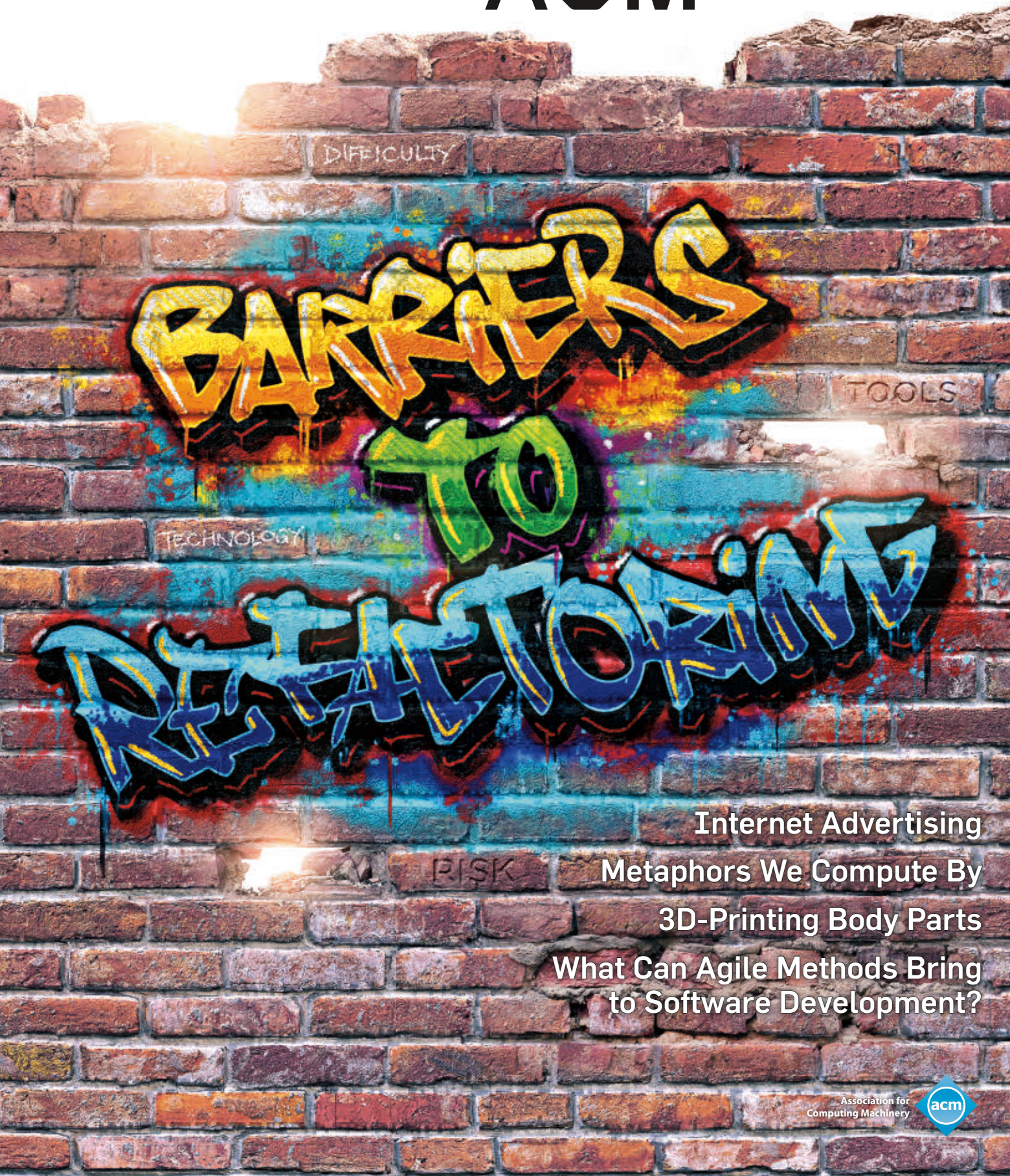


COMMUNICATIONS

CACM.ACM.ORG OF THE

ACM

10/2017 VOL.60 NO.10



Internet Advertising
Metaphors We Compute By
3D-Printing Body Parts
What Can Agile Methods Bring
to Software Development?



THE FIRST ACM TVX in ASIA

An international networking for inventing the future of TV & Media

TVX 2018 ACM International Conference on Interactive Experiences for Television and Online Video

26–28 June, 2018

Hanyang University, Seoul, Korea

–

The ACM International Conference on Interactive Experiences for TV and Online Video (ACM TVX) is the leading international conference for presenting and discussing online video and TV interaction and user experience. The conference has attracted many international researchers and practitioners from a wide range of disciplines, ranging from human-computer interaction, multimedia engineering, media design, VR/AR Technologies, to media psychology/sociology.

–

Website
<https://tvx.acm.org/2018/>

Call for Submissions

- 1–Nov–2017 SIGCHI Student Travel Grant
- 15–Nov–2017 SIGCHI Gary Marsden Student Development Fund
- 30–Nov–2017 Workshop Proposals
- 2–Feb– 2018 Full & Short Papers
- 30–Mar–2018 TVX-in-Industry, Demos, Work-in-Progress, Doctoral Consortium

Welcome submissions on

Online video/TV interaction
Online streaming experience
VR & AR technologies
Media psychology & sociology
Media contents production
Communication

**Previous
A.M. Turing Award
Recipients**

1966 A.J. Perlis
1967 Maurice Wilkes
1968 R.W. Hamming
1969 Marvin Minsky
1970 J.H. Wilkinson
1971 John McCarthy
1972 E.W. Dijkstra
1973 Charles Bachman
1974 Donald Knuth
1975 Allen Newell
1975 Herbert Simon
1976 Michael Rabin
1976 Dana Scott
1977 John Backus
1978 Robert Floyd
1979 Kenneth Iverson
1980 C.A.R Hoare
1981 Edgar Codd
1982 Stephen Cook
1983 Ken Thompson
1983 Dennis Ritchie
1984 Niklaus Wirth
1985 Richard Karp
1986 John Hopcroft
1986 Robert Tarjan
1987 John Cocke
1988 Ivan Sutherland
1989 William Kahan
1990 Fernando Corbató
1991 Robin Milner
1992 Butler Lampson
1993 Juris Hartmanis
1993 Richard Stearns
1994 Edward Feigenbaum
1994 Raj Reddy
1995 Manuel Blum
1996 Amir Pnueli
1997 Douglas Engelbart
1998 James Gray
1999 Frederick Brooks
2000 Andrew Yao
2001 Ole-Johan Dahl
2001 Kristen Nygaard
2002 Leonard Adleman
2002 Ronald Rivest
2002 Adi Shamir
2003 Alan Kay
2004 Vinton Cerf
2004 Robert Kahn
2005 Peter Naur
2006 Frances E. Allen
2007 Edmund Clarke
2007 E. Allen Emerson
2007 Joseph Sifakis
2008 Barbara Liskov
2009 Charles P. Thacker
2010 Leslie G. Valiant
2011 Judea Pearl
2012 Shafi Goldwasser
2012 Silvio Micali
2013 Leslie Lamport
2014 Michael Stonebraker
2015 Whitfield Diffie
2015 Martin Hellman
2016 Sir Tim Berners-Lee

ACM A.M. TURING AWARD NOMINATIONS SOLICITED

Nominations are invited for the 2017 ACM A.M. Turing Award. This is ACM's oldest and most prestigious award and is given to recognize contributions of a technical nature which are of lasting and major technical importance to the computing field. The award is accompanied by a prize of \$1,000,000. Financial support for the award is provided by Google Inc.

Nomination information and the online submission form are available on:
http://amturing.acm.org/call_for_nominations.cfm

Additional information on the Turing Laureates is available on:
<http://amturing.acm.org/byyear.cfm> .

The deadline for nominations/endorsements is January 15, 2018.

For additional information on ACM's award program please visit: www.acm.org/awards/



Association for
Computing Machinery

Departments

- 5 **Editor's Letter**
Computing Is a Profession
By Andrew A. Chien
-
- 7 **Cerf's Up**
Six Education
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Beyond Brute Force
-
- 12 **BLOG@CACM**
Manipulating Word Representations, and Preparing Students for Coding Jobs?
Robin K. Hill mulls an aspect of natural language processing research, while Mark Guzdial ponders why coding is taught in public schools.
-
- 33 **Calendar**
-
- 103 **Careers**

Last Byte

- 112 **Upstart Puzzles**
Partitioned Peace
By Dennis Shasha

News

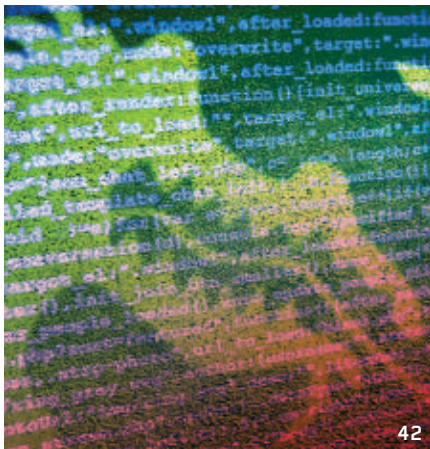


- 15 **3D-Printing Human Body Parts**
Bioprinting has generated bones, cartilage, and some muscles; hearts and livers are still years away.
By Keith Kirkpatrick
-
- 18 **Digital Hearing**
Advances in audio processing help separate the conversation from background noise.
By Don Monroe
-
- 21 **Portable Device Fears Show Power of Social Development**
How do small screens impact young minds?
By Chris Edwards

Viewpoints

- 24 **Technology Strategy and Management**
Amazon and Whole Foods: Follow the Strategy (and the Money)
Checking out the recent Amazon acquisition of Whole Foods.
By Michael A. Cusumano
-
- 27 **Inside Risks**
The Real Risks of Artificial Intelligence
Incidents from the early days of AI research are instructive in the current AI environment.
By David Lorge Parnas
-
- 32 **Economic and Business Dimensions**
FinTech Platforms and Strategy
Integrating trust and automation in finance.
By Vasant Dhar and Roger M. Stein
-
- 36 **Code Vicious**
IoT: The Internet of Terror
If it seems like the sky is falling, that's because it is.
By George V. Neville-Neil
-
- 38 **Viewpoints**
What Can Agile Methods Bring to High-Integrity Software Development?
Considering the issues and opportunities raised by Agile practices in the development of high-integrity software.
By Roderick Chapman, Neil White, and Jim Woodcock

Practice



42

- 42 **Metaphors We Compute By**
Code is a story that explains
how to solve a particular problem.
By Alvaro Videla

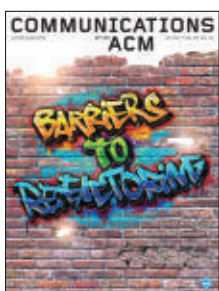
- 46 **Research for Practice: Technology
for Underserved Communities;
Personal Fabrication**
Expert-curated guides to
the best of CS research.

- 50 **Four Ways to Make CS
and IT More Immersive**
Why the Bell curve hasn't
transformed into a hockey stick.
By Thomas A. Limoncelli



Articles' development led by **acmqueue**
queue.acm.org

About the Cover:



Practitioners have turned to refactoring for years to make software code cleaner and sleeker. But studies show developers do not rely on refactoring as often as the could, and maybe should, for a variety of reasons. Our cover story (p. 54) explores the specific barriers that hinder refactoring, and suggests ways to promote its benefits. Cover illustration by Justin Metz.

Contributed Articles



54

- 54 **Barriers to Refactoring**
Developers know refactoring
improves their software, but
many find themselves unable
to do so when they want to.
*By Ewan Tempero, Tony Gorschek,
and Lefteris Angelis*



Watch the authors discuss
their work in this exclusive
Communications video.
[https://cacm.acm.org/
videos/barriers-to-
refactoring](https://cacm.acm.org/videos/barriers-to-refactoring)

- 62 **Millennials' Attitudes Toward IT
Consumerization in the Workplace**
Millennials entering the workforce
ignore the risks of using privately
owned devices on the job.
*By Heiko Gewald, Xuequn Wang,
Andy Weeger, Mahesh S. Raisinghani,
Gerald Grant, Otavio Sanchez,
and Siddhi Pittayachawan*

Review Articles

- 70 **Internet Advertising:
Technology, Ethics, and
a Serious Difference of Opinion**
Exploring the technical and ethical
issues surrounding Internet
advertising and ad blocking.
*By Stephen B. Wicker
and Kolbeinn Karlsson*



Watch the authors discuss
their work in this exclusive
Communications video.
[https://cacm.acm.org/
videos/internet-advertising](https://cacm.acm.org/videos/internet-advertising)

Research Highlights

- 80 **Technical Perspective**
**Broadening and Deepening
Query Optimization Yet
Still Making Progress**
By Jeffrey F. Naughton
- 81 **Multi-Objective Parametric
Query Optimization**
*By Immanuel Trummer
and Christoph Koch*
- 90 **Technical Perspective**
**Shedding New Light on
an Old Language Debate**
By Jeffrey S. Foster
- 91 **A Large-Scale Study
of Programming Languages and
Code Quality in GitHub**
*By Baishakhi Ray, Daryl Posnett,
Premkumar Devanbu,
and Vladimir Filkov*



Association for Computing Machinery
Advancing Computing as a Science & Profession



ACM, the world's largest educational and scientific computing society, delivers resources that advance computing as a science and profession. ACM provides the computing field's premier Digital Library and serves its members and the computing profession with leading-edge publications, conferences, and career resources.

Executive Director and CEO

Bobby Schnabel

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Darren Ramdin

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Scott E. Delman

ACM COUNCIL

President

Vicki L. Hanson

Vice-President

Cherri M. Pancake

Secretary/Treasurer

Elizabeth Churchill

Past President

Alexander L. Wolf

Chair, SGB Board

Jeanna Matthews

Co-Chairs, Publications Board

Jack Davidson and Joseph Konstan

Members-at-Large

Gabriele Anderst-Kotis; Susan Dumais; Elizabeth D. Mynatt; Pamela Samuelson; Eugene H. Spafford

SGB Council Representatives

Paul Beame; Jenna Neefe Matthews; Barbara Boucher Owens

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

Terry Coatta and Stephen Ibaraki

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Dame Professor Wendy Hall

ACM India Council

Srinivas Padmanabhuni

ACM China Council

Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson; Joseph Konstan

Board Members

Phoebe Ayers; Karin K. Breitman; Terry J. Coatta; Anne Condon; Nikil Dutt; Roch Guerrin; Chris Hankin; Carol Hutchins; Yannis Ioannidis; Michael L. Nelson; M. Tamer Ozsu; Eugene H. Spafford; Stephen N. Spencer; Alex Wade; Keith Webster; Julie R. Williamson

ACM U.S. Public Policy Office

1701 Pennsylvania Ave NW, Suite 300, Washington, DC 20006 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association

Deborah Seehorn,
Interim Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

Communications of the ACM is the leading monthly print and online magazine for the computing and information technology fields. *Communications* is recognized as the most trusted and knowledgeable source of industry information for today's computing professional. *Communications* brings its readership in-depth coverage of emerging areas of computer science, new trends in information technology, and practical applications. Industry leaders use *Communications* as a platform to present and debate various technology implications, public policies, engineering challenges, and market trends. The prestige and unmatched reputation that *Communications of the ACM* enjoys today is built upon a 50-year commitment to high-quality editorial content and a steadfast dedication to advancing the arts, sciences, and applications of information technology.

STAFF

DIRECTOR OF PUBLICATIONS

Scott E. Delman
cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Lawrence M. Fisher

Web Editor

David Roman

Rights and Permissions

Deborah Cotton

Editorial Assistant

Jade Morris

Art Director

Andriy Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Production Manager

Bernadette Shade

Advertising Sales Account Manager

Ilia Rodriguez

Columnists

David Anderson; Phillip G. Armour;
Michael Cusumano; Peter J. Denning;
Mark Guzdial; Thomas Haigh;
Leah Hoffmann; Mari Sako;
Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

Copyright permission
permissions@hq.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmhq@cacm.acm.org

Letters to the Editor

letters@cacm.acm.org

WEBSITE

<http://cacm.acm.org>

AUTHOR GUIDELINES

<http://cacm.acm.org/about-communications/author-center>

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY 10121-0701
T (212) 626-0686
F (212) 869-0481

Advertising Sales Account Manager

Ilia Rodriguez
ilia.rodriguez@hq.acm.org

Media Kit acmm mediasales@acm.org

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Andrew A. Chien
eic@cacm.acm.org

SENIOR EDITOR

Moshe Y. Vardi

NEWS

Co-Chairs

William Pulleyblank and Marc Snir

Board Members

Monica Divitini; Mei Kobayashi;
Michael Mitzenmacher; Rajeev Rastogi;
François Sillion

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;
John Leslie King; Paul Rosenbloom

Board Members

William Aspray; Stefan Bechtold;
Michael L. Best; Judith Bishop;
Stuart I. Feldman; Peter Freeman;
Mark Guzdial; Rachelle Hollander;
Richard Ladner; Carl Landwehr;
Carlos Jose Pereira de Lucena;
Beng Chin Ooi; Loren Terveen;
Marshall Van Alstyne; Jeannette Wing

PRACTICE

Chair

Stephen Bourne and Theo Schlossnagle

Board Members

Eric Allman; Samy Bahra; Peter Bailis;
Terry Coatta; Stuart Feldman; Nicole Forsgren;
Camille Fournier; Benjamin Fried;
Pat Hanrahan; Tom Killalea; Tom Limoncelli;
Kate Matsudaira; Marshall Kirk McKusick;
Erik Meijer; George Neville-Neil;
Jim Waldo; Meredith Whittaker

CONTRIBUTED ARTICLES

Co-Chairs

James Larus and Gail Murphy

Board Members

William Aiello; Robert Austin;
Elisa Bertino; Gilles Brassard; Kim Bruce;
Alan Bundy; Peter Buneman; Carl Gutwin;
Yannis Ioannidis; Gal A. Kaminka;
Karl Levitt; Igor Markov; Gail C. Murphy;
Bernhard Nebel; Lionel M. Ni; Adrian Perrig;
Sriram Rajamani; Marie-Christine Rousset;
Krishan Sabnani; Ron Shamir; Yoav Shoham;
Josep Torrellas; Michael Vitale;
Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Gregory Morrisett

Board Members

Martin Abadi; Amr El Abbadi; Sanjeev Arora;
Michael Backes; Maria-Florina Balcan;
Andrei Broder; Doug Burger; Stuart K. Card;
Jeff Chase; Jon Crowcroft; Alexei Efros;
Alon Halevy; Sven Koenig; Steve Marschner;
Tim Roughgarden; Guy Steele, Jr.;
Margaret H. Wright; Nikolai Zeldovich;
Andreas Zeller

WEB

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;
Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2017 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$269.

ACM Media Advertising Policy

Communications of the ACM and other ACM Media publications accept advertising in both print and electronic formats. All advertising in ACM Media publications is at the discretion of ACM and is intended to provide financial support for the various activities and services for ACM members. Current advertising rates can be found by visiting <http://www.acm-media.org> or by contacting ACM Media Sales at (212) 626-0686.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhq@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM*
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA

Printed in the U.S.A.



Association for
Computing Machinery





DOI:10.1145/3137136

Andrew A. Chien

Computing Is a Profession

THE NOTION OF what constitutes a *profession* has been studied extensively through exploration of the attributes of the activities, roles, and community that lead to their rise, definition, and how they achieve importance and influence and society.¹ Common among these attributes are a deep technical expertise, an essential, valued, societal contribution, and the need to adhere to high ethical and technical standards. Professions such as medicine, law, and accounting exemplify these attributes. Computing exhibits all of the attributes of a profession.

Deep Technical Expertise. Every day we witness and drive computing's rapid technical advance—new technologies, advancing sophistication, and outright new capabilities. The compounding of this continued and accelerating advance give rise to a deep technical expertise. Algorithms and systems behavioral and internal complexity are peers to the greatest complexities humanity has known in biology, society, and the universe.

Societal Recognition. Computing's evident importance to society is deep and growing—sophisticated collection and information processing underpins decision-making, logistics, and optimization industry and commerce. Web, email, and messaging platforms are the information backbone of government and commerce. Social application platforms expand these roles from official to social, insinuating computing into the core of social fabric. A world without cheap, pervasive computing of extraordinary capability is if not inconceivable at least so *distant* as to be unrecognizable. Just as modern existence without the practice of medicine or law would be unimaginable.

Necessity for High Ethical and Technical Standards. The practice of comput-

ing demands adherence to the highest technical and ethical standards, and failure has significant consequences at personal, corporate, national, and international scale. Computing applications reach into every corner. Appropriation, misuse, or just free flow of this information has demonstrably destroyed individual privacy, relationships, and careers (for example, Ashley Madison), grand corporate plans (for example, Sony), and radically changed international relations (for example, U.S. Govt OPM penetration, and Snowden releases).

With growing excitement for artificial intelligence, computing is being thrust into new societal roles (recommender, decider) and given autonomy to make decisions with life-changing human impact (personal assistant, sentencing guidelines, self-driving cars). While deep technical challenges abound, the ethical challenges, principles, and standards are even more daunting. Compared to Hanks' "rule-book" in *Bridge of Spies*, the underlying principles of a representative government, as embodied in the Constitution of the U.S., and one can only wonder if the issues are any less thorny.

So yes, computing is a profession, and we should proudly embrace the responsibility. We should welcome, educate, and mentor new generations not just as "coders" and "hackers" or programmers, but as *computing professionals*.

Computing professionals have a responsibility to practice at the state-of-the-art, and maintain their knowledge at the forefront. In addition, professionals have an obligation to share clear understanding of the technology and its implications to non-professionals, and operate in accord with professional ethics.² These are daunting goals for an individual, and so professional societies play a critical role in cultivating and supporting pro-

fessionalism. What roles specifically?

First, a professional society should be a conservator and disseminator of deep technical knowledge and expertise: championing the advance of the field by leading technologists worldwide, documenting the state-of-the-art in technology and application, and accelerating the dissemination and availability of such knowledge to computing professionals.

Second, societies develop and advocate principles for ethical technical conduct that frame the role of computing professionals, and buttress them with the stature and role of the profession in society. Examples include the articulation of best practices, intellectual challenges for the field,³ as well as address societal questions that require deep technical perspective, such as the USACM joint release on the Internet of Things.⁴

An independent professional society must transcend any individual, organization, government, or cause. Necessarily so, as technical knowledge and professional ethics must inform professional conduct, and inevitably come into conflict with personal interest, corporate interest, government or national interest, or even overt coercion.

As the leading computing professional society, the ACM seeks to fill these roles for computing!

Andrew A. Chien, EDITOR-IN-CHIEF

Andrew A. Chien is the William Eckhardt Distinguished Service Professor in the CS Department at the University of Chicago, Director of the CERES Center for Unstoppable Computing, and a Senior Scientist at Argonne National Lab.

References

1. Abbott, A. *The System of Professions: An Essay on the Division of Expert Labor*. University of Chicago Press, 1988. ISBN-13: 978-0226000695
2. ACM Code of Ethics, (1992); <https://www.acm.org/about-acm/code-of-ethics>
3. Denning, P.J. The Profession of IT. Columns in *Commun. ACM*, 2008–2017.
4. USACM. *Statement on Internet of Things Privacy and Security*, 2017; <https://www.acm.org/>

Copyright held by author.



AWARD NOMINATIONS SOLICITED

As part of its mission, ACM brings broad recognition to outstanding technical and professional achievements in computing and information technology.

ACM welcomes nominations for those who deserve recognition for their accomplishments. Please refer to the ACM Awards website at <http://awards.acm.org> for guidelines on how to nominate, lists of the members of the 2017 Award Committees, and listings of past award recipients and their citations.

Nominations are due **January 15, 2018** (formerly November 30, 2017) with the exceptions of the Doctoral Dissertation Award (due **October 31, 2017**) and the ACM – IEEE CS George Michael Memorial HPC Fellowship (due **May 1, 2018**).

A.M. Turing Award: ACM's most prestigious award recognizes contributions of a technical nature which are of lasting and major technical importance to the computing community. The award is accompanied by a prize of \$1,000,000 with financial support provided by Google.

ACM Prize in Computing (previously known as the ACM-Infosys Foundation Award in the Computing Sciences): recognizes an early-to mid-career fundamental, innovative contribution in computing that, through its depth, impact and broad implications, exemplifies the greatest achievements in the discipline. The award carries a prize of \$250,000. Financial support is provided by Infosys Ltd.

Distinguished Service Award: recognizes outstanding service contributions to the computing community as a whole.

Doctoral Dissertation Award: presented annually to the author(s) of the best doctoral dissertation(s) in computer science and engineering, and is accompanied by a prize of \$20,000. The Honorable Mention Award is accompanied by a prize totaling \$10,000. Financial support of the award is provided by Google, Inc. Winning dissertations are published in the ACM Digital Library and the ACM Books Series.

ACM – IEEE CS George Michael Memorial HPC Fellowship: honors exceptional PhD students throughout the world whose research focus is on high-performance computing applications, networking, storage, or large-scale data analysis using the most powerful computers that are currently available. The Fellowship includes a \$5,000 honorarium.

Grace Murray Hopper Award: presented to the outstanding young computer professional of the year, selected on the basis of a single recent major technical or service contribution. The candidate must have been 35 years of age or less at the time the qualifying contribution was made. A prize of \$35,000 accompanies the award. Financial support is provided by Microsoft.

Paris Kanellakis Theory and Practice Award: honors specific theoretical accomplishments that have had a significant and demonstrable effect on the practice of computing. This award is accompanied by a prize of \$10,000 and is endowed by contributions from the Kanellakis family, and financial support by ACM's SIGACT, SIGDA, SIGMOD, SIGPLAN, and the ACM SIG Project Fund, and individual contributions.

Karl V. Karlstrom Outstanding Educator Award: presented to an outstanding educator who is appointed to a recognized educational baccalaureate institution, recognized for advancing new teaching methodologies, effecting new curriculum development or expansion in computer science and engineering, or making a significant contribution to ACM's educational mission. Teachers with 10 years or less experience are given special consideration. The Karlstrom Award is accompanied by a prize of \$10,000. Financial support is provided by Pearson Education.

ACM – AAAI Allen Newell Award: presented to individuals selected for career contributions that have breadth within computer science, or that bridge computer science and other disciplines. The \$10,000 prize is provided by ACM and AAAI, and by individual contributions.

Outstanding Contribution to ACM Award: recognizes outstanding service contributions to the Association. Candidates are selected based on the value and degree of service overall.

ACM Policy Award: recognizes an individual or small group that had a significant positive impact on the formation or execution of public policy affecting computing or the computing community. This can be for education, service, or leadership in a technology position; for establishing an innovative program in policy education or advice; for building the community or a community resources in technology policy; or other notable policy activity. The biennial award is accompanied by a \$10,000 prize. The first award will be the 2017 award.

Software System Award: presented to an institution or individuals recognized for developing a software system that has had a lasting influence, reflected in contributions to concepts, in commercial acceptance, or both. A prize of \$35,000 accompanies the award with financial support provided by IBM.

ACM Athena Lecturer Award: celebrates women researchers who have made fundamental contributions to computer science. The award includes a \$25,000 honorarium provided by Google.

For SIG-specific Awards, please visit <http://awards.acm.org/sig-awards>.

Vinton G. Cerf, ACM Awards Committee Co-Chair
Insup Lee, SIG Governing Board Awards Committee Liaison

John R. White, ACM Awards Committee Co-Chair
Rosemary McGuinness, ACM Awards Committee Liaison



Vinton G. Cerf

DOI:10.1145/3134431

Six Education

How many ways can you make numbers add up to six? The obvious pairs are (1,5), (2,4), (3,3), (4,2) and (5,1). Ok, I left out (0,6) and (6,0), which are of negligible interest.

Of course, if you permit negative numbers the number of pairs is infinite: (-1,7), (-2,8), and so on. When I was growing up, we learned our addition and times tables, memorized them, and used that information to do simple arithmetic. I recently had the opportunity to join several thousand teachers in Southern California for an annual con-fab on teaching.^a I came away with a very different view of elementary and secondary education than I had going in.

For years we have tended to measure how well our students have learned by testing what they know. In the discussions at California State University, Fullerton, a different mind set emerged. Why not test what students can do? The shift in focus takes into account that we have myriad ways in the 21st century to find out what we want to know. Part of knowing how to do things is finding information when we need it. Moreover, it seems evident that we learn better not from rote memorization, but from using knowledge to do things. The Montessori schools approach learning from an experiential and exploratory perspective, for example. We learn from our mistakes sometimes more than from our successes. Certainly science tends to work that way.

The Common Core State education standards^b initiative gets at this idea by focusing on the reasons why different

methods get the same mathematical results. The point is not to use every possible way to do computation but, rather, to understand in a more fundamental way the nature of computation. Students are asked to show their work—their reasoning—as a way of determining how well they have absorbed basic concepts of mathematics, for example. Another way in which this depth of knowledge is assessed is to ask students to show they can make practical estimates of the magnitude of answers they should expect to get. This can be a quick way of ensuring the answers are in the right ballpark. For example:

$$\begin{array}{r} 197 \\ +332 \\ \hline 527 \end{array}$$

To estimate the ballpark, we could sum $200+330$ to get 530, which is a good approximation to 527 or even $200+300 = 500$ and easier to do in our heads.

In the No Child Left Behind legislation, testing knowledge was the gold standard but it led to behavior such as “teach to the test.” While this might produce good scores, it might not produce good understanding. What I think we want is to produce graduate students who know how to learn and how to find information they need from a variety of sources. This will also prove to be important as children grow into adults and experience longer working careers spanning decades. Without much doubt

these adults will need to learn new things to stay current as technology continues its relentless evolutionary pace. Adapting to change will be a career-enhancing capability.

One of the keynote speakers was Jill Biden, a teacher of teachers who, astonishingly, taught at Northern Virginia Community College while serving the U.S. as Second Lady. She referred to the ineffable pain as her stepson, Beau Biden, lost his battle with cancer and drew to mind Vice President Biden's earlier loss of his first wife, Nelia, and daughter, Naomi, in a car crash years before. In her moving account of life before, during, and after the Obama/Biden Administration she summarized: “What matters most in life is how well we will walk through fire.” This, too, was a life lesson I took away from this very thought-provoking event.

In the last couple of months I have also encountered new ways to teach mathematics in online settings. Alexander Khachatryan founded the ReasoningMind organization to create courses in mathematics for pre-K, K, elementary, and secondary school levels.^c I tried some of the lessons for the youngest children and they plainly laid a foundation for understanding the notion of sets and set membership and equivalence classes, all without using unnecessarily complex and obscure terminology. ■

^c <https://www.reasoningmind.org/>

Vinton G. Cerf is vice president and Chief Internet Evangelist at Google. He served as ACM president from 2012–2014.

Copyright held by author.

^a Better Together: California Teachers Summit 2017 (<http://cateacherssummit.com/>)

^b https://en.wikipedia.org/wiki/Common_Core_State_Standards_Initiative

Call for Nominations for ACM General Election

The ACM Nominating Committee is preparing to nominate candidates for the officers of ACM: **President, Vice-President, Secretary/Treasurer; and two Members at Large.**

Suggestions for candidates are solicited. Names should be sent by **November 5, 2017** to the Nominating Committee Chair, c/o Pat Ryan, Chief Operating Officer, ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701, USA.

With each recommendation, please include background information and names of individuals the Nominating Committee can contact for additional information if necessary.

Alexander L. Wolf is the Chair of the Nominating Committee, and the members are Karin Breitman, Judith Gal-Ezer, Rashmi Mohan, and Satoshi Matsuoka.



DOI:10.1145/3135241

Beyond Brute Force

IN THEIR ARTICLE “The Science of Brute Force” (Aug. 2017), Marjin J.H. Heule and Oliver Kullmann humorously asked whether a mathematician using brute force is really “a kind of barbaric monster.” While applying simplistic approaches to complex domains (such as image and speech processing) is inefficient, certain specific computational problems do indeed benefit from brute force. In this regard, the mathematician who uses brute force is simply functioning as a good engineer intent on solving problems efficiently.

My primary focus during my Massachusetts General Hospital fellowship (2013–2016) was analyzing the electronic health records for 314,292 patients.² To identify biomarkers associated with outcomes, my colleagues and I were initially interested in knowing the smoking status of all of them—current, past, or never—for our prediction models. Smoking status is typically documented in clinical narrative notes as free text, and, as reported throughout the literature, classification accuracy of current methods is poor.

I hypothesized that following a simple human-in-the-loop brute-force approach designed to semi-manually extract non-negated expressions could achieve better accuracy than many widely used algorithms. It was, I thought, different from common machine learning approaches that typically attempt to classify sentences or find associations between words in a series of words. Moreover, machine learning algorithms, when applied to text, rely on the assumption that any language includes an infinite number of possible expressions. In contrast, across a variety of medical conditions, we observed that clinicians tend to use the same expressions to describe patients’ conditions. For instance, the most commonly reported expressions for sleep disorders, including “poor sleep,” “sleep disruption,” and “sleeps poorly,” covered over 99%

of all possible ways to describe physician-documented insomnia.¹ We tested this approach extensively, and it played an important role in all our subsequent publications. We even extended it for additional uses (such as to extract family histories of coronary artery disease, classify patients with cancer, and improve accuracy of the Framingham risk score for patients with liver disease).

Metaphorically speaking, a rocket ship would be a potential solution if one wanted to cross the Grand Canyon, but if such a trek has no strict time-completion restriction, riding a donkey or simply walking would be more efficient, as such solutions require little maintenance or purchase of expensive fuel. Scientists confronting a computational challenge should not automatically apply readily available algorithms just because they are commonly known to be efficient. Alternatives should be considered as potentially the most efficient solution, including those that may seem to be brute force at first glance, but, as in Hans Christian Andersen’s “Ugly Duckling” story, might instead mature into the most graceful of swans.

References

1. Beam, A., Kartoun, U., Pai, J., Chatterjee, A., Fitzgerald, T., Shaw, S., and Kohane, I. Predictive modeling of physician-patient dynamics that influence sleep medication prescriptions and clinical decision making. *Scientific Reports* 9, 7 (Feb. 2017), 1–7.
2. Kartoun, U. The man who had them all. *Interactions* 24, 4 (July-Aug. 2017), 22–23.

Uri Kartoun, Cambridge, MA

For More CS Teachers Try Higher Salaries

One aspect of why more K–12 schools do not offer computer science (CS) classes not addressed by Jennifer Wang in her Viewpoint “Is the U. S. Education System Ready for CS for All?” (Aug. 2017) is how salary might lead to a lack of qualified and motivated CS teachers. People graduating from universities with CS degrees would most likely prefer to go

The most important task for any life form is to find that sweet spot where there is just the right amount of innovation.

into industry where they could earn higher salaries. They do not usually prefer to go into teaching where they might start at \$40,000 per year in public schools (depending on location) or private schools where they might earn even less. For public schools, at least in some U.S. states, they also must obtain a teacher certification in addition to a CS degree.

Colin Carlile, Devine, TX

Author Responds:

Salary and certification were beyond the scope of the study. However, many factors, including salary, influence the choice of teaching as a career and profession. Also, many potential sources of computer science teachers are constrained by certification requirements. These are excellent considerations for future research as schools seek to address the demand for computer science.

Jennifer Wang, Mountain View, CA

Helping Our Old Man Code

Philip Guo's blog "How Adults Ages 60+ Are Learning to Code" (Aug. 2017) reminded me how, in 1964, when I had been a member of ACM for just four years and my brother Michael was studying computer science at the University of Wisconsin, our father, age 60, decided he ought to learn to code if he hoped to communicate with his sons. He enrolled at Fordham University in the Bronx, NY, and took a course in Fortran, earning a B+. My brother and I discussed coding

with him for the next 25 years while he lent his business insight to helping us advance our fantastic careers.

Donald F. Costello, Lincoln, NE,
Distinguished Lecturer ACM

Infection Paths Also Needed for Internet Growth

As a developer of security in operating systems, I definitely understand that underspecifying URLs, as Vinton G. Cerf described it in his Cerf's Up column "In Praise of Under-Specification?" (Aug. 2017), allows many creative infection paths into a user's personal computer. Eric Schmidt, executive chairman of Alphabet, Inc., has declared loose specifications a necessary feature of the Internet, one that has given it the freedom to grow to the network it is today. That is somewhat like saying a swimming pool or water supply without chlorine is a feature that allows toxic algae to bloom. It is, in fact, illegal in the U.S. for a water supply to allow algae blooms. Other countries have loose water-supply specifications, thus allowing all sorts of life forms the freedom to innovate and thrive in the water supply. Some may be good for human health and well being, while others are deadly. So, too, with the loose specifications on the Internet. Douglas Hofstadter of Indiana University included a nice analogy in the prologue of his 1995 book *Fluid Concepts and Creative Analogies* describing life as living in the liquid range, between both solid and gaseous phases, neither able to support life. The most important task for any life form is to find that sweet spot where there is just the right amount of innovation; it is only in this Goldilocks zone where life is even possible. I hope humans are, in spite of all evidence to the contrary, capable of creating a society where a pleasant life is a possibility.

Tom Jones, Seattle, WA

Communications welcomes your opinion. To submit a Letter to the Editor, please limit yourself to 500 words or less, and send to letters@cacm.acm.org.

©2017 ACM 0001-0782/17/10

Coming Next Month in **COMMUNICATIONS**

**Cambits:
A Reconfigurable
Camera System**

**User Reviews for Top
Mobile Apps in the
Google Play Store**

Healthcare Robotics

**Hootsuite: In Pursuit
of Reactive Systems**

**Is There a Single Method
for the Internet
of Things?**

**Heads-Up Limit Hold'em
Poker Is Solved**

**The Heat Method for
Distance Computation**

**Social Agents: Bridging
Simulation and
Engineering**

**Pay What You Want as a
Pricing Model for Open
Access Publishing?**

**Plus the latest news about
brain-computer interfaces,
copolymers and self-assembly,
and policing online advertising.**

ACM

ON A MISSION TO SOLVE TOMORROW.

Dear Colleague,

Without computing professionals like you, the world might not know the modern operating system, digital cryptography, or smartphone technology to name an obvious few.

For over 60 years, ACM has helped computing professionals be their most creative, connect to peers, and see what's next, and inspired them to advance the profession and make a positive impact.

We believe in constantly redefining what computing can and should do.

ACM offers the resources, access and tools to invent the future. No one has a larger global network of professional peers. No one has more exclusive content. No one presents more forward-looking events. Or confers more prestigious awards. Or provides a more comprehensive learning center.

Here are just some of the ways ACM Membership will support your professional growth and keep you informed of emerging trends and technologies:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and videos through the **ACM Learning Center**
- Discounts on registration fees to ACM Special Interest Group conferences
- Subscription savings on specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

Joining ACM means you dare to be the best computing professional you can be. It means you believe in advancing the computing profession as a force for good. And it means joining your peers in your commitment to solving tomorrow's challenges.

Sincerely,



Vicki L. Hanson
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING.

JOIN ACM TODAY.

ACM is the world's largest computing society, offering benefits and resources that can advance your career and enrich your knowledge. We dare to be the best we can be, believing what we do is a force for good, and in joining together to shape the future of computing.

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

- ☐ Professional Membership: \$99 USD
- ☐ Professional Membership plus
ACM Digital Library: \$198 USD (\$99 dues + \$99 DL)
- ☐ ACM Digital Library: \$99 USD
(must be an ACM member)

ACM STUDENT MEMBERSHIP:

- ☐ Student Membership: \$19 USD
- ☐ Student Membership plus ACM Digital Library: \$42 USD
- ☐ Student Membership plus Print *CACM* Magazine: \$42 USD
- ☐ Student Membership with ACM Digital Library plus
Print *CACM* Magazine: \$62 USD

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in computing. Membership in ACM-W is open to all ACM members and is free of charge.

Priority Code: CAPP

Payment Information

Name _____

ACM Member # _____

Mailing Address _____

City/State/Province _____

ZIP/Postal Code/Country _____

Email _____

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc., in U.S. dollars or equivalent in foreign currency.

☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Total Amount Due _____

Credit Card # _____

Exp. Date _____

Signature _____

Purposes of ACM

ACM is dedicated to:

- 1) Advancing the art, science, engineering, and application of information technology
- 2) Fostering the open interchange of information to serve both professionals and the public
- 3) Promoting the highest professional and ethics standards

Return completed application to:
ACM General Post Office
P.O. Box 30777
New York, NY 10087-0777

Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

1-800-342-6626 (US & Canada)
1-212-626-0500 (Global)

Hours: 8:30AM - 4:30PM (US EST)
Fax: 212-944-1318

acmhelp@acm.org
acm.org/join/CAPP

The *Communications* Web site, <http://cacm.acm.org>, features more than a dozen bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish selected posts or excerpts.



Follow us on Twitter at <http://twitter.com/blogCACM>

DOI:10.1145/3131066

<http://cacm.acm.org/blogs/blog-cacm>

Manipulating Word Representations, and Preparing Students for Coding Jobs?

Robin K. Hill mulls an aspect of natural language processing research, while Mark Guzdial ponders why coding is taught in public schools.



Robin K. Hill
Deep Dictionary

<http://bit.ly/2tRnVZN>
June 20, 2017

Recent research in natural language processing using the program *word2vec* gives manipulations of word representations that look a lot like semantics produced by vector math. For vector calculations to produce semantics would be remarkable, indeed. The word vectors are drawn from context, big, huge context. And, at least roughly, the meaning of a word is its use (in context). Is it possible some question is begged here?

We represent words by vectors (one-dimensional arrays) of a large number of elements, the numeric values of which are determined by reading and processing a vast number of examples of context in which those words appear, and which are functions of the distances between occurrences of words in sequences. A neural network

refines the values of each element of each word vector in the training phase. The network, through iterated adjustment of the elements of the vector based on errors detected on comparison with the text corpora, produces the values in continuous space that best reflect the contextual data given. The end result is the word vector, a lengthy list of real numbers that do not seem to have any particular interpretation that would reflect properties of the thing itself, nor properties of words.

In the example provided by Mikolov et al.,¹ they "... assume relationships are present as vector offsets, so that in the embedding space, all pairs of words sharing a particular relation are related by the same constant offset." Let's use 'd' to stand for "distributed representation by vector" of a word.

d("king")	=	v1
d("male")	=	v2
d("female")	=	v2
d("queen")	=	v4

Mikolov and his colleagues find that $v1 - v2 + v3 \approx v4$ (in vector offset mathematics in the n-dimensional space). This is certainly an intriguing result, in accord with our understanding of the meanings of the four words, in which taking "king," removing the "male" aspect, and replacing it with a "female" aspect, gives "queen."

The word vectors can also capture plurality and other shades of meaning the researchers regard as syntactic. For example, the offset between singular and plural is a (learned) constant:

$d(\text{"apple"}) - d(\text{"apples"}) \approx d(\text{"families"}) - d(\text{"family"}) \approx d(\text{"cars"}) - d(\text{"car"})$

More details of *word2vec* can be found in the explanation by Xin Rong.³ It looks like, not by direct coding but by some fortuitous discovery, the system has figured out some mathematical analog for semantics. (There is no claim that individual elements of the vector are capturing features such as gender, status, or grammatical number.)

We already have a compendium of data on relationships of words to other words through contexts of use: the dictionary. The use of a word is largely given by its context. Its context can be inferred also from its dictionary definition. Most dictionaries will offer a direct or indirect connection through "king" to "ruler" or "sovereign" and "male" and through "queen" to "ruler" or "sovereign" and "female," as :

queen

The female ruler of an independent state, especially one who inherits the position by right of birth

king

The male ruler of an independent state, especially one who inherits the position by right of birth

ruler

A person exercising government or dominion

These definitions² show gender can be “factored out,” and in common usage the gender aspect of sovereigns is notable. We would expect those phenomena to show up in vast text corpora. In fact, we would expect that to show up in text corpora *because* of the dictionary entries. Since we base word use on definitions captured by the dictionary, it is natural for any graph-theoretic distance metric based on node placement to (somehow) reflect that cross-semantic structure.

Suppose that, employing the English slang terms “gal” and “guy” for male and female, the word for queen were “ruler-gal,” and for king “rulguy,” (perhaps the word for mother were “parentgal,” and for father, “parentguy”). Then the word vector offsets calculated would not appear as remarkable, the relationships exposed in the words themselves.

The system *word2vec* constructs and operates through the implicit framework of a dictionary, which gave rise to the input data to *word2vec*. How could it be otherwise? As we understand the high degree of contextual dependency of word meanings in a language, any representation of word meaning to a significant degree will reflect context, where context is its interassociation with other words.

The result is still intriguing. We have to ask how co-occurrence of words can reliably lay out semantic relationships. We might explore the aspects of semantics missing from context analysis, if any. We might (and should) ask what sort of processing of a dictionary would deliver the same sort of representations, if any.

The word vectors produced by the method of training on a huge natural text dataset, in which words are given distributed vector representations refined through associations present in the input context, reflect the cross-referential semantic compositionality of a dictionary.

That close reflection is derived from the fact that words in natural text will be arranged in accordance with dictionary definitions. The *word2vec* result is revelation of an embedded regularity.

References

1. Mikolov, T., Chen, K., Corrado, C., and Dean, J. Efficient Estimation of Word Representations in Vector Space, <https://arxiv.org/abs/1301.3781>.
2. Oxford Living Dictionary, 2017, Oxford University Press, <https://en.oxforddictionaries.com/>.
3. Rong, X. *word2vec* Parameter Learning Explained, <https://arxiv.org/abs/1411.2738>.



Mark Guzdial
Coding in Schools as
New Vocationalism:
Larry Cuban on
What Schools Are For
<http://bit.ly/2tpSgip>
July 18, 2017

Larry Cuban is an educational historian who has written before on why requiring coding in schools is a bad idea (<http://bit.ly/2uocMLQ>). Jane Margolis and Yasmine Kafai wrote an excellent response about the importance of coding in schools (<http://bit.ly/2vmi52U>). Cuban penned a three-part series about “Coding: The New Vocationalism,” likely inspired by a recent *New York Times* article about the role tech firms are having on school policy (<http://nyti.ms/2uodaKi>).

► In Part 1 (<http://bit.ly/2v2ULoo>), he describes the ‘dance’ schools have had with industry over more than 100 years, between preparing future *citizens* and preparing future *workers*.

Preparation for the workplace is not the only goal for public schooling, yet that has been the primary purpose for most reformers over the past three decades. A century ago, reformers also elevated workplace preparation as the overarching purpose for tax-supported public schools.

In the new vocationalism, Cuban sees schools have been tied to economic growth and the needs of information-age society. He sees coding advocates blending the roles of school in preparing citizens and school as preparing workers by arguing that computing is necessary for modern society.

► In Part 2 (<http://bit.ly/2vmjCG8>), he points out any education reform faces the reality of what teachers *know* and what they will actually *do* in the classroom. He draws on efforts in the 1950s and 1960s, and uses the story of Logo to explain how reformers get it wrong.

The lessons to be learned from earlier school reformers are straightforward.

► *Build teacher capabilities in content and skills since both determine to what degree, if any, a policy gets past the classroom door.*

► *With or without enhanced capabilities and expertise, teachers will adapt policies aimed at altering how and what they teach to the contours of the classrooms in which they teach. If policymakers hate teacher fingerprints over innovations, if they seek fidelity in putting desired reforms into practice, they wish for the impossible.*

► *Ignoring both of the above lessons ends up with incomplete implementation of desired policies and sorely disappointed school reformers.*

► In Part 3 (<http://bit.ly/2wpo9o8>), he returns to the question of what school is for. He describes successful reform as a collaboration between top-down designers and policy-makers and bottom-up teachers. He describes a successful model for reform that created “work circles” of researchers and teachers (at Northwestern University) to achieve the goals of the researchers’ curriculum by adapting it with the help of the teachers.

Cuban is not necessarily against teaching computing in schools; he says it doesn’t make sense to impose it as a mandate from industry. More importantly, he offers a path forward: *mutual adaptation* of curricular goals, between designers and teachers.

Mutual adaptation can benefit teachers and students. While this is only one study of four teachers wrestling with teaching a science unit, it is suggestive of what can occur.

Will similar efforts involve teachers and make the process of mutual adaptation work for both teachers and students? I have yet to read of such initiatives as districts and states mandating computer science courses and requiring young children to learn to code. Repeating the errors of the past and letting mutual adaptation roll out thoughtlessly has been the pattern thus far. The “New Vocationalism,” displaying a narrowed purpose for tax-supported public schools, marches on unimpeded.

Robin K. Hill is an adjunct professor in the Department of Philosophy at the University of Wyoming. Mark Guzdial is a professor in the College of Computing at Georgia Institute of Technology.

Copyright held by authors.

SEC 2017

The 2nd ACM/IEEE Symposium on Edge Computing

San Jose, CA, October 12-14, 2017

<http://acm-ieee-sec.org/2017/>

Register Now

Keynote Speaker



Prof. Mahadev Satyanarayanan
Carnegie Mellon Univ.



Dr. Pablo Rodriguez
CEO of Telefonica Alpha

General Chair

Junshan Zhang, Arizona State Univ.

Program Chairs

Mung Chiang, Purdue Univ.

Bruce Maggs, Akamai/Duke Univ.

Steering Committee

Victor Bahl, Microsoft Research

Flavio Bonomi, Nebbiolo Technologies, Inc

Rong N. Chang, IBM Research

Dejan Milojevic, HP Labs

Michael Rabinovich, Case Western Reserve Univ.

Weisong Shi (Chair), Wayne State Univ.

Tao Zhang, Cisco

Conference Sponsorship



Association for
Computing Machinery



Advancing Technology
for Humanity



TCI: Technical Committee on the Internet

Platinum Sponsors



Gold Sponsors



3D-Printing Human Body Parts

Bioprinting has generated bones, cartilage, and some muscles; hearts and livers are still years away.

THE ADVENT OF three-dimensional (3D) printing is already yielding benefits in many fields by improving the speed and efficiency of product development, prototyping, and manufacturing, while also enabling true on-off customization to suit individual needs. While this is certainly a boon for manufacturers, 3D printing holds even greater promise when considering the possibilities for creating body parts or tissues to replace or repair organs or limbs that have worn out, become damaged, or have been lost to due injury or disease.

Indeed, significant ongoing research is being conducted at the university level into the use of 3D printing to create a variety of replacement parts for aspects of the human anatomy.

Unlike standard two-dimensional printers that affix toner to sheets of paper, 3D printers create objects by putting down a layer of a powdered substrate, heating it to harden it, and then repeating the process, layer by layer, building the item up to create a completed object.

The notion of using 3D printing (also known as additive manufacturing) to replace parts of the human



Three-dimensionally printed prosthetic replacements for a human nose and ear.

body, a process known as bioprinting, was born out of a process initially developed 20 years ago, when surgeon Anthony Atala and his team at Boston Children's Hospital started to build novel tissues for regenerative medicine by hand. In this process, cells

were harvested from patients, then painstakingly layered on top of one another to create handmade "scaffolds"; then the cells would grow and multiply, eventually producing functional tissues, including skin, bladder, and cartilage.

The process was far too slow and labor-intensive to be practical, so Atala's team, which had relocated to the Wake Forest Institute for Regenerative Medicine in Winston-Salem, NC, reconfigured standard desktop inkjet printers so they could use the inkjets to shoot out cells. The idea behind bioprinting is to use a small nozzle to precisely control the placement of cells, and the biomaterials around the cells, to form a structural framework, while also creating a construct that can deliver nutrients to ensure the cells' growth.

One of the key considerations with bioprinting is the need to maintain a 3D structure. The liquid "inks," which contain cells and structural biomaterial such as collagen, must gel as quickly as they are deposited, to maintain their form. The printers, with multiple nozzles, allow for the simultaneous printing of multiple cell types, structural biomaterials, and other chemicals. This, of course, requires significant pre-planning calculations to be made focused on cell density, type, and proportions of other supporting biomaterial prior to the printing, although according to researchers, much of this can be modeled by observing existing tissue taken from patients, and then replicating that structure with biomaterial.

More recently, Wake Forest University's Sang Jin Lee and his team have been working on the development of an integrated tissue-organ printer that can fabricate stable, human-scale tissue constructs of any shape. To ensure the shape of the tissue construct is correct, clinical imaging data is fed into a computer model of the damaged or missing tissue, and that model data is then translated into a program that controls the motions of the printer nozzles, which dispense cells to specific locations. The printer also incorporates micro-channels into the constructs, which allow the precise diffusion of nutrients to the printed cells, ensuring their survival.

The key benefit to using 3D printers is the ability to create complex parts relatively quickly and efficiently. "Using 3D-printing techniques, we can easily build complex structures," Lee says, noting that in a clinical situation with real patients, "you need more

complex tissue and more control; 3D printing can provide that."

The Wake Forrest University team has successfully fabricated mandible and calvarial (skullcap) bone, cartilage, and skeletal muscle, and expects to conduct additional clinical trials over the next three to five years to demonstrate the effectiveness of its process.

Researchers on Long Island, NY, have addressed 3D bioprinting in a different manner, using relatively inexpensive consumer-grade 3D printers to print replacement body parts, including bone, cartilage, and a trachea. Indeed, Todd Goldstein, a 3D bioprinting researcher and director of Northwell Ventures 3D Printing Laboratory, Manhasset, NY, has printed tracheas made of living cells, using a Makerbot Replicator 2, a 3D printer that has been modified to print biomaterial, at a cost of under \$2,000.

Goldstein modified the Makerbot printer with a syringe that dispenses bio-ink, a material made of living cells with a viscous consistency. The nozzle that comes standard on the machine is used to emit an organic plastic called PLA (polylactic acid), which creates the scaffold of the trachea. The material costs just a few dollars, and is biodegradable in the body so only the body part is left after implantation is completed.

Goldstein says his team's work is mostly focused on bone, cartilage, and trachea printing, due to the current limitations surrounding the efficient and effective printing of constructs

that support blood vessels.

"The biggest issue on the whole right now is blood supply," Goldstein says. "You can't make the constructs too large without running into the issue of having a necrotic core (an area of dead cells) in the middle. Just like our body needs blood circulating all the time, so do these living tissues."

Studies have shown the printed tracheal tissue was just as strong as the tracheal tissue found naturally in mammals, and since the success of printed tracheas, the team has gone on to print other types of body parts, including bone and cartilage. These parts are particularly strong candidates for 3D printing because each individual will require uniquely sized parts. Furthermore, because bone, cartilage, and tendons don't require a blood supply, they can be implanted without the possibility of rejection.

Indeed, that is why the printing of more-complex body parts is still likely several years away. Creating pieces of soft tissue to mend organs such as the liver, kidney, and heart, can be quite challenging to bioprint, given the need to support constant, consistent blood flow.

"The biggest challenge is creating blood vessels that will work immediately," Goldstein says. "When you bioprint cells, if they don't have a blood supply, they'll be dead in five hours or less."

Goldstein acknowledges that at present, "there isn't a good solution for that right now. That's why a lot of research is going into 3D-printing small capillaries and a vascular network that can support larger 3D-printed constructs, like the liver, kidney, or heart."

Work is progressing on creating structures that can support blood flow. Adam Feinberg, associate professor of materials science and biomedical engineering at Carnegie Mellon's College of Engineering, has developed a new, less-expensive way of 3D printing biological soft materials, including coronary arteries and embryonic hearts. Feinberg also uses modified consumer-grade 3D printers that use open-source software to control and fine-tune their print parameters.

What makes Feinberg's work

The printing of complex body parts is likely still several years away. The soft tissue required to mend organs can be challenging to bioprint.

unique is his approach to printing soft materials, such as arteries, which need to be supported while being printed so they don't fall apart. Feinberg and his team developed a method of printing soft materials inside a support bath material, which creates one gel inside of another, thereby allowing the structure to be printed while also being supported in the correct position.

"My lab has developed a number of novel 3D-printing approaches, where we put these gels inside of a support gel," Feinberg says. "We fill up a container with one gel, then drop a needle in the middle, and then print another gel with the [structure] material."

Another key challenge that Feinberg highlights is, "What you actually print is not what you're going to want in the end." He explains that the initial living tissue that is printed will need to be cultured and grown into the final structure that will be implanted into a human. "That is a total unknown right now, because we don't have good computer models for that process yet."

Feinberg contrasts bioprinting body parts with printing static parts used in manufacturing, where it is possible to model how a part will wear or function over time, based on previously known data from materials science and engineering work. "In bioprinting, the cells are highly variable, the biomaterial is also highly variable, and the interaction of the cells and material over time is just not understood at all," Feinberg says. "That's a huge area that needs improvement."

As a result, it is unlikely that 3D printing of biomaterial will become commonplace within the next few years, due to the significant amount of research and medical trials that need to be conducted.

There also are challenges around maintaining sufficiently high quality control, for both safety and health issues, as well as in keeping regulatory bodies, like the U.S. Food and Drug Administration, happy. Researchers agree additional tools are needed for assessing the quality of printed constructs so they are developing properly, and in the case of living tissue, sourcing and acquiring cells on which to model 3D-printed cells can be time consuming, expensive, or both.

It is unlikely 3D printing of biomaterial will become commonplace over the next few years, due to the research and medical trials still needed.

"There's broader challenges to bioprinting," Feinberg says. "We need a lot of cells, and some cells are just hard to come by, or very costly to get. For example, heart cells don't actually divide normally, and there's no real stem cell population to generate them in the adult heart." While Feinberg says it is now possible to use embryonic cells to grow new ones, it remains very costly to do so.

Wake Forest's Lee agrees, noting that the real challenges lie around printing complex organs. "We are trying to mimic original tissue as much as possible," Lee says. However, "for complex organs like the kidney and heart, we need better imaging scans and more cell types," Lee says. "Unlike simple tissue like bone or cartilage, the kidney requires more than 30 cell types." **C**

Further Reading

3D Bioprinting System for Human-Scale Tissue Constructs: <http://www.nature.com/nbt/journal/v34/n3/full/nbt.3413.html>

Northwell Health Wins Award for 3D Bioprinting: <https://www.northwell.edu/about/news/northwell-health-announces-3d-bioprinting-winner-medical-innovation-contest>

Video: Adam Feinberg of Carnegie Mellon University discusses 3D bioprinting: https://www.youtube.com/watch?v=Zfl_tFdt2D4&feature=youtu.be

Keith Kirkpatrick is principal of 4K Research & Consulting, LLC, based in Lynbrook, NY.

© 2017 ACM 0001-0782/17/10 \$15.00

ACM Member News

DREAMING OF A SECURE INTERNET



"I was always interested in many things, but somehow my mind was like a Petri dish for computer

science ideas," says Adrian Perrig, professor of computer science at ETH Zurich in Switzerland. "I happen to be good at it."

Perrig displayed an early affinity for programming. "On a borrowed Commodore 64, I wrote my first program, which was a health indicator I saw in a book I was reading."

Realizing he liked programming and was good at it, he decided to study computer science. Perrig earned his undergraduate degree in computer engineering from École polytechnique fédérale de Lausanne (EPFL) in Switzerland, and a Ph.D. in computer science from Carnegie Mellon University.

He launched his career as a professor at Carnegie Mellon, where he remained for 11 years. Perrig then returned to Switzerland as a professor for ETH Zurich, where he now leads the network security group.

Perrig focuses on networking and systems security, specifically in the design of a secure next-generation Internet architecture. "My dream has been to develop a secure Internet," he says. To that end, he started SCION, an acronym for Scalability, Control, and Isolation On Next-Generation Networks, in 2009.

The hard work has paid off, as the SCION network is now functional. "Our routers are in ISPs, and a global test bed is being set up to make use of this secure infrastructure," Adrian adds.

SCION has been Perrig's sole focus over the past few years, and will remain so, he says. "If you spread yourself too thin, it is hard to have an impact."

—John Delaney

Digital Hearing

Advances in audio processing help separate the conversation from background noise.

LIKE MANY OTHER technologies, hearing aids have improved rapidly over the last two decades.

Digital audio processing has enabled more sophisticated and personalized features, and users report higher satisfaction. Still, in noisy social situations such as cocktail parties, hearing-impaired users must work hard to track the conversation of companions, just as someone with normal hearing may struggle to follow a lively conference call.

Researchers and companies continue to advance the technology by understanding how listeners process a complex “auditory scene,” which requires more than just amplifying sounds. “Restoring audibility is simply not enough,” said Christine Jones, director of the Phonak Audiology Research Center in Warrenville, IL. “There’s a loss of fidelity that prevents individuals from having full restoration of function even when the sound becomes audible again.”

Research also has shown that people process sound in different ways, so the “best” hearing aid technology will differ, too. Even as researchers struggle to measure these differences, companies are providing new ways to address them and to develop fitting procedures and user controls that work best for each individual.

Digital Revolution

Digital signal processors (DSPs) were first used in hearing aids in 1996. In the ensuing years, almost all hearing devices adopted them because of their flexibility in providing programmable features.

For example, simple amplification makes soft sounds perceptible, but loud sounds uncomfortable. DSPs make it easy to adjust both the gain (sound boost) and the dynamic-range compression to match an individual’s hearing loss at different frequencies.



The Earlens Light-Driven Hearing Aid converts sounds into pulses of light, which activate a lens on the eardrum.

DSPs also suppress the feedback that arises from the close proximity of microphone and speaker, and can reduce distracting background noise, such as wind.

In recent years, hearing aid companies also have introduced frequency-lowering technology. Patients with extreme hearing loss at high frequencies can benefit when sounds are shifted to frequencies where they still have some function, said Brent Edwards, CTO of Earlens in Menlo Park, CA. “It’s a distortion, but if you do it effectively, you can get some phonemes still through the damaged auditory system.”

Hearing researchers have explored other processing strategies over the years, but many of them remain in the

labs because of the severe design constraints posed by small device size and the need for low power consumption to preserve battery life. In addition, processing must be done with very limited delay, especially because the processed signal must coexist with sound passing through bypass channels that avoid uncomfortable ear-canal blockage.

In special situations, engineers can dramatically improve listeners’ experiences by incorporating external devices. For example, asking conversation partners to clip a microphone on their collar greatly reduces background noise, and combining the output of several microphones on a table can enhance a particular speaker. External loudspeakers, as at a business, may

also provide a more compelling spatial perception of sound than in-ear microphones. Yet everyday use still requires small, low-power devices.

One useful type of external device that many people already carry is the smartphone. A Bluetooth connection to a phone app gives users a richer and more intuitive interface, for example, even with a low data rate. A more bandwidth-intensive approach offloads some signal processing to the phone, with its larger battery and higher processing power, but long delays are unacceptable.

Auditory Scenes

Although increased processing power has clearly benefited hearing aid technology, designs must extend beyond electrical engineering to encompass the complex and idiosyncratic ways that people process and interpret sounds. A particularly influential concept is auditory scene analysis, which posits that people perceive sound not as frequencies but as individual sources, each emitting waveforms with similar attributes. In this view, hearing aids should preserve or accentuate the cues that let a listener identify and attend to sources of interest while disregarding others.

As a step in that direction, hearing aid companies often categorize the sound environment and vary the processing algorithms accordingly. “We have at least 12 different attributes we look at,” including the frequency spectrum, the modulation characteristics, and the angle of the sound deduced by comparing two microphones, said Francis Kuk of the Office of Research in Clinical Amplification (ORCA-USA) of Widex, in Lisle, IL.

Based on these parameters, Widex will predict that the environment contains music, speech in a quiet environment, speech in a train station, or other settings. The inferred scene then determines the default parameters of the hearing aid, such as the noise reduction for a noisy environment.

The auditory scene can also be used to modify beamforming, which wirelessly combines audio signals from both earpieces. The technique can enhance sounds coming from straight ahead, for example, because they arrive at the ears in sync. “It’s one of our

“As you develop new technologies, you have to develop new ways of measuring its benefit. That’s part of what engineers don’t understand about hearing loss.”

most effective features,” says Jones, but “we have different levels of beamforming depending on the scene.” As background noise increases, for example, the effective beam is made increasingly narrow. But if a car scene is detected, the beamforming is turned off, because speakers may well be to the side or behind the listener.

Individual Strategies

There are limits to such automated selection of algorithms, however, due not only to varying preferences of individuals, but to differences in how their brains process sound.

One feature that helps distinguish sources in an auditory scene is the overall modulation of the auditory signal, Kuk said. Some people, particularly those with both hearing impairment and cognitive limitations, “are more reliant on the temporal waveform” to identify a source, for example when different frequencies get louder at the same time. Hearing aids that respond rapidly to protect users from sudden loud sounds distort the temporal waveform, making it difficult for these listeners to use this important cue.

The effort and attention required to decipher speech can be exhausting for hearing-impaired people, but that is not easy to assess. “We used to say speech understanding and sound-quality assessment were the two measures of what a technology was doing. Now we can add a third dimension, and that’s cognitive impact,” said Edwards. “Some technologies may not improve speech understanding that much, but reduce cognitive load.”

Edwards said the “hottest topic in hearing science right now” is called “hidden hearing loss,” in which patients lose some of the normal nerve connections. “It’s like losing half the pixels on your high-def TV,” he said, but it does not affect the audibility of pure tones.

The standard test of detection threshold “is not telling you, say in a restaurant, how salient the representation of the auditory scene is,” Edwards warned. “As you develop new technologies, you have to develop new ways of measuring its benefit,” he said. “That’s part of what engineers don’t understand about hearing loss. There’s a lot about the individual patient that we can’t tell from the diagnostics.”

Fitting the Listener

Fine-tuning settings is already a key aspect of “fitting” of a new hearing aid by a trained audiologist, which contributes to the devices’ price. Edwards compares this process to improving a TV picture by yelling instructions to a technician who can’t see the picture. “What people are now doing is developing systems where the patient can make adjustments themselves,” he said. “There has to be a balance between diagnostic-driven fitting by a professional and this sort of self-adjustment.”

“We’ve done a really good job now of creating algorithms that work on average, based on the degree and configuration of the hearing loss,” said Jones, who has practiced as an audiologist, including fitting hearing aids to pediatric patients with limited communication skills. “But two patients who present with the same hearing loss could have very different levels of functional auditory behavior. There’s still room to do some fine-tuning based on their own personal capabilities.”

In light of the high price of current hearing aids (which can cost as much as \$4,000 per unit), there is increasing pressure to open the market to cheaper over-the-counter (OTC) devices. Lowered regulatory requirements, advocates suggest, might encourage companies like Dolby and Bose to leverage their expertise in entertainment audio for personal devices.

“I support creating an OTC category, as long as it doesn’t affect the traditional distribution and provision of hearing aids,” Edwards said. Kuk cau-

tioned such over-the-counter products are unlikely to have the advanced capabilities of medically-approved hearing aids like those his company offers, and may instead resemble the generic reading glasses sold in drug stores.

“As an audiologist, I believe that for many people with hearing loss, taking something off the shelf will not do the best job,” Jones agreed, although patients with limited impairment may find such devices helpful. “There are still people that benefit from the ability of an audiologist to more closely fine-tune the amplification and the behavior of the hearing instruments to both their hearing loss and also their lifestyle needs.”

Distinguishing Sources

In spite of steady advances in hearing aids, users still struggle to distinguish voices in a hubbub of other voices, said DeLiang Wang of Ohio State University. “Every hearing-aid company recognizes that the cocktail-party problem is not solved—and needs to be solved.”

In the laboratory, Wang and his colleagues divide the audio stream into 20ms-long segments in various frequency bands, and ask whether each

segment contributes to the speech of interest or can be disregarded as noise. A similar binary classification and masking is used to make MP3 audio encoding more efficient. “Once you have formulated it as a classification problem, then modern machine learning techniques can be utilized; in our case, deep neural networks,” Wang said. Lab testing shows that masking the “noisy” time-frequency segments significantly improves listeners’ speech comprehension.

If research like this can be exploited in practical devices, it could work with listeners’ brains to help extract specific sources from a complex audio scene. But when speech is also the noise, Jones said, “there’s no acoustic separation, other than direction and maybe level, between your target and your interference. That is the challenge.”

In spite of these aspirations for the future, “the satisfaction rate for hearing aids today is much, much higher than several years ago, and especially before digital hearing aids,” Kuk said. Industry-sponsored surveys show that “almost 90% of patients who wear today’s hearing aids are satisfied,” he explained. “Having said that, satisfaction is a mov-

ing target. As technology improves, people’s expectations for what a satisfying hearing aid is also increases.” **C**

Further Reading

Abrams, H., and Kihm, J.
An Introduction to MarkeTrak IX:
A New Baseline for the Hearing Aid Market,
Hearing Review, June 2015,
<http://bit.ly/2sKl3cL>

Edwards, B.
A Model of Auditory-Cognitive Processing
and Relevance to Clinical Applicability,
Ear & Hearing July/August 2016,
<http://bit.ly/2rX9DpQ>

Rönnberg, J., et al.
The Ease of Language Understanding (ELU)
model: theoretical, empirical, and clinical
advances, *Front. Syst. Neurosci.*, 13 July
2013, <http://bit.ly/2rtSD93>

Wang, D.
Deep Learning Reinvents the Hearing Aid,
IEEE Spectrum, December 6, 2016,
<http://bit.ly/2qYmhQm>

Stix, G.
How Hearing Works [Video],
Scientific American, August 1, 2016,
<http://bit.ly/2qQsbE0>

Don Monroe is a science and technology writer based in Boston, MA.

© 2017 ACM 0001-0782/17/10 \$15.00

Milestones

ACM Honors Researchers For Innovations

ACM recently named the recipients of four prestigious technical awards, selected by their peers for making significant contributions. They were among those honored at the ACM Awards Banquet in June.

Mahadev Satyanarayanan, Michael L. Kazar, Robert N. Sidebotham, David A. Nichols, Michael J. West, John H. Howard, Alfred Z. Spector, and Sherri M. Nichols were named recipients of the ACM Software System Award for developing the Andrew File System, the first distributed file system designed for tens of thousands of machines, and pioneering the use of scalable, secure, and ubiquitous access to shared file data. The award is presented to an institution or individual(s) recognized for developing a software system that has had a lasting influence, reflected in contributions to concepts, in commercial acceptance, or both.

Jeffrey Heer was named recipient of the ACM Grace Murray Hopper Award, for developing visualization languages that have changed the way people build and interact with charts and graphs across the Web. The award is given to the outstanding young computer professional of the year, selected on the basis of a single recent major technical or service contribution.

Amos Fiat and Moni Naor received the ACM Paris Kanellakis Theory and Practice Award for the development of broadcast encryption and traitor tracing systems. The award honors theoretical accomplishments that have had a significant, demonstrable effect on the practice of computing.

Jitendra Malik was selected to receive the ACM-AAAI Allen Newell Award for seminal

contributions to computer vision that have led the field in image segmentation and object category recognition. The award is presented to an individual selected for career contributions that have breadth within computer science, or that bridge computer science and other disciplines.

MIT GRADUATE EARNS DOCTORAL DISSERTATION AWARD

Haitham Hassanieh, assistant professor in the Department of Electrical and Computer Engineering and the Department of Computer Science at the University of Illinois at Urbana-Champaign, recently was awarded the ACM 2016 Doctoral Dissertation Award, presented annually to the author(s) of the best doctoral dissertation(s) in computer science and engineering.

Hassanieh developed highly efficient algorithms for computing the Sparse Fourier Transform, and demonstrated their applicability in many domains including networks, graphics, medical imaging, and biochemistry.

Hassanieh’s Sparse Fourier Transform algorithm was chosen by *MIT Technology Review* as one of the top 10 breakthrough technologies of 2012. He has also been recognized with the Sprowls Award for Best Dissertation in Computer Science, and with the Best Paper Award from SIGCOMM, ACM’s professional forum for the discussion of topics in the field of communications and computer networks, including technical design and engineering, regulation and operations, and the social implications of computer networking.

Portable Device Fears Show Power of Social Development

How do small screens impact young minds?

LAST YEAR, THE American Academy of Pediatrics (AAP) updated its guidelines on how much access children should have to electronic devices amid growing concerns among parents of the effects of electronic media. Yet extrapolation of the evidence linking television and behavior may obscure potentially more subtle and diverse effects. Recent developments in work with interactive devices represent an increased understanding of how children learn and the importance of social interaction.

Concerns over the mental effects of electronic devices have been largely driven by fears that the prevalence of conditions such as attention-deficit/hyperactivity disorder (ADHD) seem to follow their adoption. In 2011, the U.S. Centers for Disease Control and Prevention (CDC) reported an increase of 33% in ADHD prevalence among children from 1997 to 2008. A 2016 follow-up study by the CDC found the increase continued to 2012, but then began to fall through 2015 among children of poorer families, although that reduction was not reflected in wealthier homes.

From 1999 onward, a number of studies looked at heavy television use by young children and identified possible effects on their development. Dr. Dimitri Christakis, director of the Center for Child Health, Behavior, and Development at Seattle Children's Hospital, says: "We found that the more television children watched at age three, the more likely they were to have attention problems." The fast-changing images and sounds in many television programs that are made to capture the attention of young children "condition the mind to a reality that doesn't exist," he notes.

The question is whether such a link extends to portable devices, says Dr.



Danielle Erkoboni-Wilbur, a pediatrician at St. Christopher's Hospital for Children in Philadelphia. "When those studies on behavior were done, technology just meant television. There is nothing that's formally in the literature linking those outcomes to portable digital media."

A 2015 study by pediatrician Dr. Hilda Kabali and colleagues at the Einstein Medical Center in Philadelphia found much of the average child's time with portable devices is spent watching online TV, rather than using interactive applications. However, as less passive experiences become more common over time, they may have different effects on children's behavior.

Says Christakis, "I think the touch-screen interactive experience that began with the iPad is, for the lack of a better word, a transformative technology. It's the interactivity that makes it different. With traditional media, a child never thinks or says 'I did it': it's a completely passive experience. But it's so gratifying to make something happen by touching an object on the screen."

"What we are finding in our lab is that these devices command attention much better than other things. It can make it more difficult for parents to interact with their children," Christakis adds. "I tend to think of the effects being mediated through two different pathways. One is the direct pathway, which is the actual content. Interactive media could lead to the same kind of overstimulation as fast-paced TV, although being interactive, it means the child can control the pacing in a way that isn't possible with television."

"There is also the indirect pathway, which works through displacement. This is about what could they be doing that they aren't, whether it's singing, reading, or going outside to play. Even if someone developed the perfect app that was perfectly paced and shown to be beneficial, if children used that app eight hours a day, we would recognize that behavior as being a problem," Christakis adds.

Christakis is concerned about the addictiveness of applications on tablets and smartphones, and the potential for them to eat too far into the child's daytime activities. "I tell parents to limit time on interactive devices to no more than 30 minutes. People ask, 'How do I know?' I came up with that limit when we looked at what people have done in the past, using techniques such as time diaries. Children in the pre-iPad days typically spent a maximum of 20–30 minutes a day with a particular toy, but they will often spend much more time than that with an iPad game; there is something very different about the experience."

"The makers of many of these games design them to be addictive. We live in an attentional economy, so apps writers are often trying to get people drawn in and get people addicted," Christakis adds.

Alexis Hiniker, a Ph.D. candidate in Human Centered Design and Engineering at the University of Washington, says parents quickly detect over-addictive apps. “A lot of the families we talked to, they said ‘we tried X but it was so hard to get them to put it away that we had to take it away’. They had to go cold turkey. So we find apps that are super-grabby are not successful, at least when it comes to younger children.”

Taking the device away inevitably causes tension. Hiniker’s own work involves finding ways to design apps for children that make it easier to set limits on usage time without causing tantrums when device time has to stop. “Our approach was to look at what is helpful to children in acquiring self-regulation. We found if children have explicit opportunities to make plans, they will take more ownership over their behavior,” she explains.

The Plan & Play system trialed by Hiniker lets children work out how long they want to spend on an activity before starting. “As they are playing, we surface that information back to them and remind them of what their plan was.

“In terms of practical considerations for industry, I would love to see design for things like parental controls shift to include these ideas, [and] move away from lockout mechanisms that simply force children to do other things.”

Erkoboni-Wilbur says a focus on designing devices and apps for children can avoid the potential for harm and make them support better development. “There are a variety of futures in front of us. We can see how easy it is for technology to isolate folks, but we do see pointers in research as to what causes great outcomes; what we can do to optimize technology to improve developmental outcomes.”

Allison Druin, a professor in the College of Information Studies at the University of Maryland, College Park, agrees: “Technology, as with any tool in a person’s life, is either going to amplify the challenges or support the strengths. We have the situation where autistic kids finally feel comfortable socially because they have this technology bridge, but there is also the potential to amplify inattention.”

One underappreciated limitation of screen-based devices that has emerged in studies is their two-dimensional (2D) nature. That realization is helping to bolster research on how children learn. Research by developmental psychologist Rachel Barr and colleagues at Georgetown University revealed infants find it more difficult than older children and adults to translate learning from the 2D space to three dimensions.

“We call it the transfer deficit. A very common problem in learning in general is when we have take something from one context to another. What you are faced with on the screen is different to what you are faced with in the real world,” Barr says. “We need to understand that this transfer deficit is important. Children can seem to be digital natives when they use these devices so easily, but it’s actually more cognitively demanding to learn from them.”

What makes it easier for children to learn is what Barr calls the “social scaffold.” The Georgetown researchers showed this effect in an experiment based on a jigsaw-like puzzle. Researchers demonstrated the game to half the children. The other half only viewed a “ghost version”: a puzzle that assembles itself onscreen without human intervention. When given the tablet to solve the puzzle, the young children who only saw the ghost version were often unable to complete the exercise. “We gave them the touchscreen and they were baffled, but if someone showed them first, they were really good at it.”

Barr points to a 2014 study performed by Northwestern University doctoral candidate Courtney Blackwell (now a research assistant professor of medical social sciences at Northwestern’s Feinberg School of Medicine) with kindergarten-age children at three schools in Chicago as another example of the social scaffold in action, this time with children working together. At the beginning of the school year, children at one location received an iPad, another school did not supply them at all, and a third gave iPads to pairs of students. Those who shared devices scored higher on literacy tests than the other two groups.

“This human ability is really quite an amazing one that we have. We can take information from someone and link

information really quite rapidly,” Barr says. “There is something about a person helping you that is different, but the device itself can often seem so compelling. I wonder if maybe parents sometimes think maybe the device can teach them better,” Barr adds.

Researchers such as Christakis insist parents should resist the temptation to believe the device will be better at teaching and to be confident in what they can bring to the situation. “One has to keep in mind that our brains have evolved over millennia; they are contingent on social interaction. Every time we are interacting with a child, we are laying down new synapses and making new connections in the brain.”

Erkoboni-Wilbur concludes: “The message that we are focusing on is that technology can be a social experience. We need to guide it away from being an isolated type of interaction. We are really encouraging parents to sit down together and learn with technology. Parents and their children, teachers and children, children and their peers; experience it as a social medium, and enhance that social interaction.” **C**

Further Reading

Kabali, H. K., Irigoyen, M. M., Nunez-Davis, R., Budacki, J. G., Mohanty, S. H., Leister, K. P., and Bonner Jr, R. L.

Exposure and use of mobile media devices by young children, *Pediatrics*, 136, 1044–1050 (2015)

Chassiakos, Y.R., Radesky, J., Christakis, D., Moreno, M.A., and Cross, C.

Children and Adolescents and Digital Media, *American Academy of Pediatrics Technical Report*.

Barr, R.

Memory Constraints on Infant Learning From Picture Books, Television, and Touchscreens, *Child Development Perspectives* (2013)

Hiniker, A., Lee, B., Sobel, K., and Choe, E.K. Plan and Play: Supporting Intentional Media Use in Early Childhood, *Proceedings of the 16th Annual ACM Conference on Interaction Design and Children (IDC '17)*

Blackwell, C. K.

iPads in Kindergarten? Investigating the effects of tablet computers on student achievement, *Proceedings of the International Communication Association* (2015).

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

Introducing *ACM Transactions on Human-Robot Interaction*

Now accepting submissions to ACM THRI

In January 2018, the *Journal of Human-Robot Interaction* (JHRI) will become an ACM publication and be rebranded as the *ACM Transactions on Human-Robot Interaction* (THRI).

Founded in 2012, the *Journal of HRI* has been serving as the premier peer-reviewed interdisciplinary journal in the field.

Since that time, the human-robot interaction field has experienced substantial growth. Research findings at the intersection of robotics, human-computer interaction, artificial intelligence, haptics, and natural language processing have been responsible for important discoveries and breakthrough technologies across many industries.

THRI now joins the ACM portfolio of highly respected journals. It will continue to be open access, fostering the widest possible readership of HRI research and information. All issues will be available on the ACM Digital Library.



Editors-in-Chief Odest Chadwicke Jenkins of the University of Michigan and Selma Šabanović of Indiana University plan to expand the scope of the publication, adding a new section on mechanical HRI to the existing sections on computational, social/behavioral, and design-related scholarship in HRI.

The inaugural issue of the rebranded *ACM Transactions on Human-Robot Interaction* is planned for March 2018.

To submit, go to <https://mc.manuscriptcentral.com/thri>





DOI:10.1145/3132722

Michael A. Cusumano

Technology Strategy and Management

Amazon and Whole Foods: Follow the Strategy (and the Money)

Checking out the recent Amazon acquisition of Whole Foods.

IN JUNE 2017, Amazon announced it would acquire Whole Foods, the national grocery chain with nearly 500 stores, for \$13.7 billion in cash. This is less than Whole Foods' 2016 revenues of \$15.7 billion, which included \$507 million in net profits (3.2% of revenues). Whole Foods became a takeover target because of declining sales and profits as well as increasing competition in the cutthroat supermarket business.¹ By comparison, Amazon's 2016 revenues were \$136 billion, up 27% over 2015, with operating profits of \$4.1 billion and net profits of \$2.4 billion (1.8% of revenues). Amazon's market value of around \$480 billion is 3.5 times last year's revenues. It is buying Whole Foods for less than one times its last year's revenues (0.85%). Given that Whole Foods is actually more profitable than Amazon, which plows most of its potential profits into

new business development and expansion, and since Amazon does not have to dilute its stock, this seems like a great deal for CEO Jeff Bezos. But is it? And what does this acquisition say about Amazon's strategy?

Launched by Bezos in 1994 as a pioneering online bookstore, Amazon

Unlike Google and other Internet platforms, Amazon was never a completely virtual business.

has always made deft use of physical assets as well as the Internet. It soon offered two million or more titles—far more than actual bookstores were able to stock. Later in the 1990s, Amazon added a platform service that linked buyers who wanted less-popular books with third-party sellers that held the inventory, a business now called Amazon Marketplace. Unlike Google and other Internet platforms, Amazon was never a completely virtual business. Bezos operated out of a warehouse that intercepted shipments from distributors and then re-sent the books to customers. The company also began to buy large numbers of best-seller books and stock them so Amazon could benefit from scale economies in purchasing and earn premiums from delivering the books quickly.

Eventually, Amazon expanded to other popular items suitable for sale over the Internet, ranging from elec-



tronic goods to digital content (music and videos) to clothing. Amazon now sells some two million items.⁴ A decade ago it also launched Amazon Web Services (AWS) to sell excess computing and storage capacity from its massive data centers. In addition, Bezos experimented with his own products such as the successful Kindle tablet for e-books, the failed Fire smartphone, and the intriguing Echo/Alexa digital assistant and speaker device. More recently, Amazon opened a bookstore in Seattle to complement the online store, and it is experimenting with new technology (Amazon Go) that eliminates cashiers.

Until now, Amazon's pieces fit together pretty well. Now the company is moving deeper into the low-margin grocery business. We have to wonder why.

Amazon's financials are not particularly strong compared to other leading technology companies and Internet

platforms, and mix products and services with varying profit rates. In 2016, sales of products accounted for 70% of Amazon's revenues.² Sales of services (mostly from selling third-party goods on Amazon Marketplace, revenues from AWS, some allocations from Amazon Prime membership fees, and some advertising) accounted for the other 30%. AWS produced only \$12.2 billion or 9% of revenues (compared to 5% in 2014) but a whopping \$3.1 billion (74%) of operating profits. By contrast, Amazon lost \$1.3 billion on overseas sales of \$44 billion, nearly one-third of revenues. It is also estimated that Amazon loses up to \$2 billion per year on Prime Memberships.⁶ With 70- to 80-million members, Prime generated \$6.8 billion in 2016 revenue.⁵ Prime offers free shipping, a lot of free digital content, and other benefits for a \$99 annual fee or less in some cases. The free services are costly but encourage

customer loyalty and seem to drive long-term sales growth.

Bezos' interest in groceries goes back a decade, when he launched Amazon Fresh. Why? *Volume*. The annual grocery business in the U.S. alone was worth some \$1.3 trillion in 2016, with Walmart (which has 4,500 stores) having the largest market share at 18%.⁷ Amazon and Whole Foods together would have a market share of about 3.5%.¹¹ Amazon has figured out how to sell perishable and non-perishable groceries via the Web. It is not clear the company has figured out how to do this at a profit. Amazon can learn from Whole Foods how to sell groceries to upscale customers, but this market is shrinking.

Many customers are the same, however, so Amazon can try cross-marketing. About 62% of Whole Foods customers are Amazon Prime members.⁹ Perhaps Amazon can convert the 38%

COMMUNICATIONS APPS

Access the latest issue, past issues, **BLOG@CACM**, News, and more.



Available for iPad, iPhone, and Android



Available for iOS, Android, and Windows

<http://cacm.acm.org/about-communications/mobile-apps>



Association for Computing Machinery

Bezos' goal does not seem to be profit, at least not in the short term.

of Whole Foods customers who are not yet Prime members. Perhaps Amazon can expand the reach of Whole Foods through its Web presence and online delivery services. Perhaps the Whole Foods network of stores and warehouses can help Amazon in storing products or handling returns from Web purchases, or showcasing products for sale.

Are there significant synergies between selling books and selling lettuce, or between warehousing groceries and other items? Maybe, but maybe not. Synergies on paper are always difficult to realize in acquisitions. That is why, in many studies, at least two-thirds of acquisitions fail. In this case, Amazon will find that economies of scale and scope in the grocery business are not like the digital world. There may even be some economic penalties with expansion or automation if Amazon cannot maintain the quality and service that are hallmarks of Whole Foods.

Another argument against the acquisition is that the more sectors into which Amazon diversifies or integrates vertically, the more it resembles a conglomerate with no particular specialization or competitive advantage.⁸ Perhaps Amazon should use the \$13.7 billion to expand AWS. This is Amazon's most profitable business, but it is subject to intense competition from Microsoft, Google, IBM, and other companies.

Although the Federal Trade Commission approved the takeover in August, lawyers and analysts have raised long-term anti-trust concerns.⁶ Amazon disrupted the competition by pricing physical books, e-books, electronics, diapers, digital media, and other goods low—and sometimes below cost—to gain market share. Then

it benefitted from increasing scale economies and positive feedback loops associated with digital platforms and network effects (see “The Evolution of Platform Thinking,” *Communications*, January 2010). That is, the more goods and services Amazon sells, the more customers it has, and the more likely it becomes that more buyers and sellers will use Amazon, especially if competitors vanish or falter. With weak or no competitors, Amazon is also free to raise prices.

Bezos' goal does not seem to be profit, at least not in the short term. Rather, he seems intent on expanding Amazon's reach. As noted in the title of a 2013 book on the company, Bezos wants Amazon to be “The Everything Store.”¹⁰ Whole Foods fits this strategy of retail expansion. It also fits Bezos' apparent belief that, to expand its reach, Amazon needs to increase its physical presence. Why physical? *Follow the money*. Nearly 44% of U.S. consumers go first to Amazon when they want to make a purchase.⁶ However, the vast majority of purchases we make (about 92%) still occur in brick-and-mortar stores, not over the Internet.³ The value of this acquisition will be in how effectively and broadly Amazon is able to utilize the new physical platform it is buying. **C**

References

1. Abrams, R. and Creswell, J. Amazon deal for Whole Foods starts a supermarket war. *The Wall Street Journal* (June 16, 2017).
2. Amazon.com, Inc. *Form 10-K Annual Report*.
3. Bray, H. Whole Foods deal fits into Amazon's plan to offer one-stop shopping. *Boston Globe* (June 16, 2017).
4. Elgan, M. This is why Amazon will open physical bookstores. *Computerworld* (Feb. 8, 2016).
5. Gustafson, K. Amazon hints at one of its best kept secrets: How many prime members it has. *CNBC.com* (Feb. 17, 2017).
6. Khan, L. Amazon's antitrust paradox. *Yale Law Journal* (Jan. 2017), 751–752.
7. Russell, K. and Seshagiri, A. Amazon is trying to do (and sell) everything. *The New York Times* (June 16, 2017).
8. Sorkin, A. Conglomerates didn't die. They look like Amazon. *The New York Times* (June 19, 2017).
9. Stevens, L. and Haddon, H. Big prize in Amazon-Whole Foods deal: Data. *The Wall Street Journal* (June 20, 2017).
10. Stone, B. *The Everything Store: Jeff Bezos and the Age of Amazon*. Little, Brown, NY, 2013.
11. Wingfield, N. and de la Merced, M. Amazon to buy Whole Foods for 13.4 billion. *The New York Times* (June 16, 2017).

Michael A. Cusumano (cusumano@mit.edu) is a professor at the MIT Sloan School of Management and founding director of the Tokyo Entrepreneurship and Innovation Center at Tokyo University of Science.

The author thanks Annabelle Gawer, Xiaohua Yang, and David Yoffie for their very helpful comments on earlier versions of this column.

Copyright held by author.

Inside Risks

The Real Risks of Artificial Intelligence

Incidents from the early days of AI research are instructive in the current AI environment.

THE VAST INCREASE in speed, memory capacity, and communications ability allows today's computers to do things that were unthinkable when I started programming six decades ago. Then, computers were primarily used for numerical calculations; today, they process text, images, and sound recordings. Then, it was an accomplishment to write a program that played chess badly but correctly. Today's computers have the power to compete with the best human players.

The incredible capacity of today's computing systems allows some purveyors to describe them as having “artificial intelligence” (AI). They claim that AI is used in washing machines, the “personal assistants” in our mobile devices, self-driving cars, and the giant computers that beat human champions at complex games.

Remarkably, those who use the term “artificial intelligence” have not defined that term. I first heard the term more than 50 years ago and have yet to hear a scientific definition. Even now, some AI experts say that defining AI is a difficult (and important) question—one that they are working on. “Artificial intelligence” remains a buzzword, a word that many think they understand but nobody can define.

Recently, there has been growing alarm about the potential dangers of artificial intelligence. Famous giants of the commercial and scientific world have expressed concern that AI will

Application of AI methods can lead to devices and systems that are untrustworthy and sometimes dangerous.

eventually make people superfluous. Experts have predicted AI will even replace specialized professionals such as lawyers. A Microsoft researcher recently made headlines saying, “As artificial intelligence becomes more powerful, people need to make sure it's not used by authoritarian regimes to centralize power and target certain populations.”^a

Automation has radically transformed our society, and will continue to do so, but my concerns about “artificial intelligence” are different. Application of AI methods can lead to devices and systems that are untrustworthy and sometimes dangerous.

An Early Introduction to AI

As a student at Carnegie Mellon University (CMU),^b I learned about “artificial intelligence” from some of the field's founders. My teachers were clever but took a cavalier, “Try it and fix it,” attitude toward programming. I missed the disciplined approach to problem solving that I had learned as a student of physics, electrical engineering, and mathematics. Science and engineering classes stressed careful (measurement-based) definitions; the AI lectures used vague concepts with unmeasurable attributes. My engineering teachers showed me how to use physics and mathematics to thoroughly analyze problems and products; my AI teachers relied almost entirely on intuition.

I distinguished three types of AI research:

- ▶ building programs that imitate human behavior in order to understand human thinking;
- ▶ building programs that play games well; and
- ▶ showing that practical computerized products can use the methods that humans use.

Computerized models can help researchers understand brain function. However, as illustrated by Joseph Weizenbaum,² a model may duplicate the “black-box” behavior of some mechanism without describing that mechanism.

^a Interview with Kate Crawford in *The Guardian*, March 13, 2017.

^b CMU was then known as Carnegie Institute of Technology.

INTERACTIONS



ACM's *Interactions* magazine explores critical relationships between people and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of interaction design.

Our readers represent a growing community of practice that is of increasing and vital global importance.



To learn more about us, visit our award-winning website <http://interactions.acm.org>

Follow us on Facebook and Twitter



To subscribe: <http://www.acm.org/subscribe>



Writing game-playing programs is harmless and builds capabilities. However, I am very concerned by the proposal that practical products should apply human methods. Imitating humans is rarely the best way for a computer to perform a task. Imitating humans may result in programs that are untrustworthy and dangerous.

To explain my reservations about AI, this column discusses incidents from the early days of AI research. Though the stories are old, the lessons they teach us remain relevant today.

Heuristic Programming

AI researchers sometimes describe their approach as “heuristic programming.” An early CMU Ph.D. thesis defined a *heuristic program* as one that “does not always get the right answer.” Heuristic programs are based on “rules of thumb,” that is, rules based on experience but not supported by theory.^c

“Heuristic” is not a desirable attribute of software. People can use rules of thumb safely because, when rules suggest doing something stupid, most people won’t do it. Computers execute their programs unquestioningly; they should be controlled by programs that can be demonstrated to behave correctly in any situation that might arise. The domain of applicability of a program should be clearly documented. Truly trustworthy programs warn their users whenever they are applied outside that domain.

Heuristics can be safely used in a program if:

- ▶ The specification allows several acceptable solutions and the heuristic is used either to select one of them or to determine the presentation order.
- ▶ The heuristic is intended to speed up a program that conducts a search that will either find a solution or establish that there is no solution.

In other situations, heuristic programming is untrustworthy programming.

What Alan Turing Really Said

Alan Turing is sometimes called the “Father of AI” because of a 1950 paper,

^c Those who write heuristic programs rarely characterize the set of conditions under which the program would produce an incorrect result. See the section “An AI System for Constructing Parsers.”

“Computing Machinery and Intelligence.”¹ It is frequently claimed that, in that paper, Turing proposed a test for machine intelligence.

Those who believe that Turing proposed a test for machine intelligence should read that paper. Turing understood that science requires agreement on how to measure the properties being discussed. Turing rejected “Can machines think?” as an unscientific question because there was no measurement-based definition of “think.” That question is not one that a scientist should try to answer.

Turing wrote: “If the meaning of the words ‘machine’ and ‘think’ are to be found by examining how they are commonly used it is difficult to escape the conclusion that the meaning and the answer to the question, ‘Can machines think?’ is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.”

Turing’s proposed replacement question was defined by an experiment. He described a game (the imitation game) in which a human and a machine would answer questions and observers would attempt to use those answers to identify the machine. If questioners could not reliably identify the machine, that machine passed the test.

Turing never represented his replacement question as equivalent to “Can machines think?” He wrote, “The original question, ‘Can machines think?’ I believe to be too meaningless to deserve discussion.” A meaningless question cannot be equivalent to a scientific one.

Most of Turing’s paper was not about either machine intelligence or thinking; it discussed how to test whether or not a machine had some well-specified property. He also speculated about when we might have a machine that would pass his test and demolished many arguments that might be used to assert that no machine could ever pass his test. He did not try to design a machine that would pass his test; there is no indication that he thought that would be useful.

Joseph Weizenbaum’s Eliza

Anyone interested in the Turing Test should study the work of the late MIT

professor Joseph Weizenbaum.³ In the mid-1960s, he created Eliza, a program that imitated a practitioner of Rogerian psychotherapy.^d Eliza had interesting conversations with users. Some “patients” believed they were dealing with a person. Others knew that Eliza was a machine but still wanted to consult it. Nobody who examined Eliza’s code would consider the program to be intelligent. It had no information about the topics it discussed and did not deduce anything from facts that it was given. Some believed Weizenbaum was seriously attempting to create intelligence by creating a program that could pass Turing’s test. However, in talks and conversations, Weizenbaum emphasized that was never his goal. On the contrary, by creating a program that clearly was not “intelligent” but could pass as human, he showed that Turing’s test was not an intelligence test.

Robert Dupchak’s Penny-Matcher

Around 1964,^e the late Robert Dupchak, a CMU graduate student, built a small box that played the game of “penny matching.”^f His box beat us consistently. Consequently, we thought it must be very intelligent.

It was Dupchak who was intelligent—not his machine. The machine only remembered past moves by its opponent and assumed that patterns would repeat. Like Weizenbaum, Dupchak demonstrated that a computer could appear smart without actually being intelligent. He also demonstrated that anyone who knew what was inside his box would defeat it. In a serious application, it would be dangerous to depend on such software.

Character Recognition

A popular topic in early AI research and courses was the character recognition problem. The goal was to write programs that could identify hand-drawn or printed characters. This task, which most of us perform effortlessly, is dif-

ficult for computers. The optical character recognition software that I use to recognize characters on a scanned printed page frequently errs. The fact that character recognition is easy for humans but still difficult for computers is used to try to keep programs from logging on to Internet sites. For example, the website may display^g a CAPTCHA as shown here



and require the user to type “s m w m.” This technique works well because the character recognition problem has not been solved.

Early AI experts taught us to design character recognition programs by interviewing human readers. For example, readers might be asked how they distinguished an “8” from a “B.” Consistently, the rules they proposed failed when implemented and tested. People could do the job but could not explain how.

Modern software for character recognition is based on restricting the fonts that will be used and analyzing the properties of the characters in those fonts. Most humans can read a text, in a new font without studying its characteristics, but machines often cannot. The best solution to this problem is to avoid it. For texts created on a computer, both a human-readable image and a machine-readable string are available. Character recognition is not needed.

An AI System for Constructing Parsers

As a new professor, I made appointments with three famous colleagues to ask how to recognize a good topic for my students’ Ph.D theses. The late Alan Perlis, the first recipient of ACM’s prestigious Turing Award, gave the best answer. Without looking up from his work, he said, “Dave, you’ll know one when you see it. I’m busy; get out of here!” Two other Turing Award winners, the late Allen Newell and the late Robert Floyd, met with me. Separately, both said that while they could not answer my question directly, they would discuss both a good thesis and a bad one. Interestingly,

Newell’s example of a good thesis was Floyd’s example of a bad one.

The disputed thesis presented an AI program that would generate parsers^h from grammars. Newell considered it good because it demonstrated that AI could solve practical problems. Floyd, a pioneer in the field of parsing, explained that nobody could tell him what class of grammars the AI parser generator could handle, and he could prove that that class was smaller than the class of languages that could be handled by previously known mathematical techniques. In short, while the AI system appeared to be useful, it was inferior to systems that did not use heuristic methods. Bob Floyd taught me that an AI program may seem impressive but come out poorly when compared to math-based approaches.

An AI System that “Understood” Drawings and Text

A 1967 AI Ph.D thesis described a program that purportedly “understood” both natural language text and pictures. Using a light pen and a graphics display,ⁱ a user could draw geometric figures. Using the keyboard, users could ask questions about the drawing. For example, one could ask “Is there a triangle inside a rectangle?” When the author demonstrated it, the program appeared to “understand” both the pictures and the questions. As a member of the examining committee, I read the thesis and asked to try it myself. The system used heuristics that did not always work. I repeatedly input examples that caused the system to fail. In production use, the system would have been completely untrustworthy.

The work had been supervised by another Turing Award recipient, Herbert Simon, whose reaction to my observing that the system did not work was, “The system was not designed for antagonistic users.” Experience has shown that computer systems must be prepared for users to be careless and, sometimes, antagonistic. The techniques used in that thesis would not be acceptable in any commercial

^h Parsers, an essential component of compilers, divide a program into its constituent parts. Before Floyd’s work, parsers were created by humans. Floyd’s algorithm automatically generated parsers for a large class of languages.

ⁱ Advanced hardware for the time.

^d Practitioners of Rogerian psychotherapy echo the patient’s words in their responses.

^e Dupchak’s accidental death prevented publication of his work. I cannot give a precise date.

^f Penny-matching is a two-player game. Each player uses a coin to make a head or tail choice. One player wins if both pick the same face; the other wins if the choices are different.

^g This example was found in Wikipedia.

product. If heuristics are used in critical applications, legal liability will be a serious problem.

An AI Assembly-Line Assistant

An assembly line could run faster after tool-handling assistants were hired: Whenever workers finished using a tool, they tossed it in a box; when a tool was needed, the assistant retrieved it for the workers.

A top research lab was contracted to replace the human assistants with robots. This proved unexpectedly difficult. The best computer vision algorithms could not find the desired tool in the heap. Eventually, the problem was changed. Instead of tossing the tool into the box, assemblers handed it to the robot, which put it in the box. The robot remembered where the tool was and could retrieve it easily. The AI controlled assistant could not imitate the human but could do more. It is wiser to modify the problem than to accept a heuristic solution.

“Artificial Intelligence” in German^j

When AI was young, a German psychology researcher visited pioneer AI researchers Seymour Papert and Marvin Minsky (both now deceased) at MIT. He asked how to say “artificial intelligence” in German because he found the literal translation (Künstliche Intelligenz^k) meaningless.

Neither researcher spoke German. However, they invited him to an AI conference, predicting that he would know the answer after hearing the talks. Afterward, he announced that the translation was “natürliche Dummheit” (natural stupidity) because AI researchers violated basic rules of psychology research. He said that psychology researchers do not generally ask subjects how they solve a problem because the answers might not be accurate; if they do ask, they do not trust the answers. In contrast, AI researchers were asking chess players how they decide on their next move and then writing programs based on the player’s answers.

^j I cannot warrant the truth of this story; it was related to me as true, but I was not present for the events. I include it because it contains an important lesson.

^k Current terminology in German.

Artificial Neural Networks

Another approach to AI is based on modeling the brain. Brains are a network of units called neurons. Some researchers try to produce AI by imitating the structure of a brain. They create models of neurons and use them to simulate neural networks. Artificial neural networks can perform simple tasks but cannot do anything that cannot be done by conventional computers. Generally, conventional programs are more efficient. Several experiments have shown that conventional mathematical algorithms outperform neural networks. There is intuitive appeal to constructing an artificial brain based on a model of a biological brain, but no reason to believe this is a practical way to solve problems.

The Usefulness of Physics and Mathematics

A researcher presented a paper on using AI for image processing to an audience that included experts in radar signal processing. They observed that the program used special cases of widely used signal-processing algorithms and asked “What is new in your work?” The speaker, unaware of techniques used in signal processing, replied, “My methods are new in AI.” AI researchers are often so obsessed with imitating human beings that they ignore practical approaches to a problem.

A study of building temperature-control systems compared an AI approach with one developed by experienced engineers. The AI program monitored individual rooms and turned on the cooling/heating as needed. The engineers used a heat-flow model that included the building’s orientation, the amount of sunlight hitting sections of the building, the

**Learning is not magic,
it is the use of data
collected during use
to improve future
performance.**

heat absorption/loss characteristics of the building, and so on. Using this model, which allowed their system to anticipate needs, and the ability to pump heat from one part of the building to another, they designed a system that reduced temperature fluctuations and was more energy efficient.

Humans do not have the measurement and calculation ability that is available to a modern computer system; a system that imitates people won’t do as well as one based on physical models and modern sensors.

Humans solve complex physics problems all the time. For example, running is complex. Runners maintain balance intuitively but have no idea how they do it. A solution to a control problem should be based on physical laws, and mathematics, not mimicking people. Computers can rapidly search complex spaces completely; people cannot. For example, a human who wants to drive to a previously unvisited location is likely to modify a route to a previously visited nearby place. Today’s navigation devices can obtain the latest data and calculate a route from scratch and often find better routes than a human would.

Machine Learning

Another approach to creating artificial intelligence is to construct programs that have minimal initial capability but improve their performance during use. This is called *machine learning*. This approach is not new. Alan Turing speculated about building a program with the capabilities of a child that would be taught as a child is taught.¹ Learning is not magic; it is the use of data collected during use to improve future performance. That requires no “intelligence.” Robert Dupchak’s simple penny-matching machine used data about an opponent’s behavior and appeared to “learn.” Use of anthropomorphic terms obscures the actual mechanism.

Building programs that “learn” seems easier than analyzing the actual problem, but the programs may be untrustworthy. Programs that “learn” often exhibit the weaknesses of “hill-climbing”¹ algorithms; they can miss the best

¹ *Hill-climbing algorithms* are analogous to hikers who always walk uphill. They may end up at the top of a foothill far below the mountain peak.

solution. They may also err because of incomplete or biased experience. Learning can be viewed as a restricted form of statistical classification, mathematics that is well developed. Machine-learning algorithms are heuristic and may fail in unusual situations.

Robot Ethics

When people view computers as thinking or sentient beings, ethical issues arise. Ethicists traditionally asked if the use of some device would be ethical; Now, many people discuss our ethical obligations to AIs and whether AIs will treat us ethically. Sometimes ethicists posit situations in which AI must choose between two actions with unpleasant consequences, and ask what the device should do. Because people in the same situation would have the same issues, these dilemmas were discussed long before computers existed. Others discuss whether we are allowed to damage an AI. These questions distract us from the real question, “Is the machine trustworthy enough to be used?”

Wordplay

The AI research community exploits the way that words change meaning: the community’s use of the word “robot” is an example. “Robot” began as a Czech word in Karel Čapek’s play, *R. U. R.* (Rossum’s Universal Robots). Čapek’s robots were humanoids, almost indistinguishable from human beings, and acted like humans. If “robot” is used with this meaning, building robots is challenging. However, the word “robot” is now used in connection with vacuum cleaners, bomb-disposal devices, flying drones, and basic factory automation. Many claim to be building robots even though devices remotely like Karel Čapek’s are nowhere in sight. This wordplay adds an aura of wizardry and distracts us from examining the actual mechanism to see if it is trustworthy. Today’s “robots” are machines that can, and should, be evaluated as such. When discussing AI, it is important to demand precise definitions.

AI: Creating Illusions

Alan Perlis referred to AI researchers as “illusionists” because they try to create the illusion of intelligence. He argued they should be considered stage magicians rather than scientists. Dupchak

Whenever developers talk about AI, ask questions. Although “AI” has no generally accepted definition, it may mean something specific to them.

and Weizenbaum demonstrated it is easy to create the illusion of intelligence.

We do not want computer systems that perform tricks; we need trustworthy tools. Trustworthy systems must be based on sound mathematics and science, not heuristics or illusionist’s tricks.

Conclusion

Whenever developers talk about AI, ask questions. Although “AI” has no generally accepted definition, it may mean something specific to them. The term “AI” obscures the actual mechanism but, while it often hides sloppy and untrustworthy methods, it might be concealing a sound mechanism. An AI might be using sound logic with accurate information, or it could be applying statistical inference using data of doubtful provenance. It might be a well-structured algorithm that can be shown to work correctly, or it could be a set of heuristics with unknown limitations. We cannot trust a device unless we know how it works.

AI methods are least risky when it is acceptable to get an incorrect result or no result at all. If you are prepared to accept “I don’t understand” or an irrelevant answer from a “personal assistant,” AI is harmless. If the response is important, be hesitant about using AI.

Some AI programs almost always work and are dangerous because we learn to depend on them. A failure may go undetected; even if failures are detected, users may not be prepared to proceed without the device.

Do not be misled by demonstrations: they are often misleading because the demonstrator avoids any situations where the “AI” fails. Computers can do many things better than people. Humans have evolved through a sequence of slight improvements that need not lead to an optimal design. “Natural” methods have evolved to use our limited sensors and actuators. Modern computer systems use powerful sensors and remote actuators, and can apply mathematical methods that are not practical for humans. It seems very unlikely that human methods are the best methods for computers.

When Alan Turing rejected “Can machines think?” as unscientific, and described a different question to illustrate what he meant by “scientific,” he was right but misled us. Researchers working on his “replacement question” are wasting their time and, very often, public resources. We don’t need machines that simulate people. We need machines that do things that people can’t do, won’t do, or don’t do well.

Instead of asking “Can a computer win Turing’s imitation game?” we should be studying more specific questions such as “Can a computer system safely control the speed of a car when following another car?” There are many interesting, useful, and scientific questions about computer capabilities. “Can machines think?” and “Is this program intelligent?” are not among them.

Verifiable algorithms are preferable to heuristics. Devices that use heuristics to create the illusion of intelligence present a risk we should not accept. ■

References

1. Turing, A.M. Computing machinery and intelligence. *Mind* 59 (1950), 433–460.
2. Weizenbaum, J. Automating psychotherapy. *ACM Forum Letter to the Editor. Commun. ACM* 17, 7 (July 1974), 425; doi: 10.1145/361011.361081.
3. Weizenbaum, J. ELIZA—A computer program for the study of natural language communication between man and machine. *Commun. ACM* 9, 1 (Jan. 1966), 36–45; doi: 10.1145/365153.365168.

David Lorge Parnas works for Middle Road Software, Inc., in Ottawa, Canada. He is Professor Emeritus at McMaster University in Canada and the University of Limerick in Ireland.

Lillian Chik-Parnas, Nancy Leveson, Peter Denning, and Peter Neumann offered helpful suggestions about earlier drafts of this column.

Copyright held by author.

Economic and Business Dimensions FinTech Platforms and Strategy

Integrating trust and automation in finance.

TEN YEARS FROM now, will Google and Amazon play more of a role in managing your investment portfolio than Fidelity? Finance has traditionally been about trust. We trust banks to hold our money and give it back to us when we want it; and we trust brokerage firms to buy the securities we want at market prices and to debit and credit our accounts accordingly. Because trust is so important, we have historically held banks and financial firms to much higher standards of compliance and control than other businesses. Financial institutions are required to follow well-defined processes with oversight and failsafe plans aimed at minimizing risk and maximizing public trust. These processes have traditionally involved humans, even as they have been increasingly augmented with technology over time.

But the last 20 years have seen the emergence of a fundamentally new and different phenomenon enabled by Internet connectivity and access: the “platform” business. In platform businesses, many complex processes, including compliance and checks-and-balance procedures, are performed securely by machines. Along with this platform phenomenon, newer generations of humans have co-evolved to be more comfortable trusting machines



to make or support decisions. It would have seemed unthinkable 10 years ago to imagine trusting autonomous driving vehicles to take over the wheel, or trust robots to perform surgery on us. But we are increasingly doing just that in a growing number of real-world areas. Broadly, we seem comfortable trusting machine-made decisions in domains in which the risks are accept-

able: mistakes are relatively infrequent and their consequences do not exceed a reasonable tolerance threshold.¹

The question is, will future investors trust FinTech platforms to the degree that previous generations have trusted traditional banks? Conversely, what will it take for FinTech platforms to be trusted sufficiently by future generations?

FinTech and Platforms

To answer this type of question, we first need to define what we mean by “FinTech” and “platform,” and identify the “platform opportunities” in finance. We define FinTech as: “Financial sector innovations involving technology-enabled business models that can facilitate disintermediation; revolutionize how existing firms create and deliver products and services; address privacy, regulatory and law-enforcement challenges; provide new gateways for entrepreneurship; and seed opportunities for inclusive growth.”^a

We define a *platform* as an entity that provides “A nexus of operational and business rules; integrated technology architecture and engines; and channel access that facilitates exchange between two or more interdependent groups, usually consumers and producers.”^{3,5}

Platforms have a number of important properties, but today’s “complete” Internet platforms always have three essential components:

1. They are “open,” allowing easy participation;

2. They implement key business and operational processes, some of which typically exhibit network effects that increase in value as participation increases; and

3. They implement these business processes automatically using enabling technology (which may also capture and generate vast amounts of data that enhances the value of the platform).⁶

Internet businesses are increasingly structured as platforms, and in most markets, a few such platforms tend to dominate. Facebook has dominated social networking with 1.8 billion users;^b Amazon has a 65% market share in online books; Google’s global search market share is 77%;^c Uber claims more than 80% of U.S. market share in its market.^d And so on. Complete platforms such as these become increasingly difficult to dislodge once they are established. As a result, the allure of market dominance provides a significant incentive to establish a complete platform.

a See <http://bit.ly/2dZnJxN>

b See <http://bit.ly/2daz7Yr>

c See <http://bit.ly/1MYzonY>

d See <http://bit.ly/2wMvqiQ>

Internet businesses are increasingly structured as platforms, and in most markets, a few such platforms tend to dominate.

FinTech Platforms

Because of historical factors and regulation, platforms in finance have tended to be “incomplete” in that they lack one or more of the three essential components.² Exchanges are examples of some of the earliest “members only” incomplete platforms—the Royal Exchange of London (opened in 1571),^e and the Osaka Rice Exchange (*Dōjima kome ichiba*, established in 1697) are early instances of these.⁴ Modern exchanges have replaced manual process with machines and added process sophistication, but are still generally accessible only to members in order to minimize risk to the platform. Similarly, clearinghouses and payment platforms with restricted access are incomplete.

Competition among platforms often arises from differentiation of some component. For example, electronic exchanges, the evolutionary successors to the Royal Exchange of London and the Osaka Rice Exchange, are complete platforms, allowing a much broader set of investors to trade assets such as foreign exchange, equities, or bonds directly with each other and without human brokers. Dozens of e-exchanges have arisen in the last two decades, each designed to meet some kind of specialized need, such as the ability to transact large sizes or to provide incentives to specific types of liquidity takers or makers, and so on. Liquidnet is an example of a complete platform, well suited for institutional trading involving execution of large blocks of assets that are often illiquid. Historically these were handled through brokers to

e See <http://bit.ly/2wMvqiQ>

Calendar of Events

September 29–October 2

MEMOCODE '17: 15th ACM-IEEE International Conference on Formal Methods and Models for System Design, Vienna, Austria, Contact: Jean-Pierre Talpin, Email: talpin@irisa.fr

October 1–4

ICTIR '17: ACM SIGIR International Conference on the Theory of Information Retrieval, Amsterdam, Netherlands, Sponsored: ACM/SIG, Contact: Evangelos Kanoulas, Email: ekanoulas@gmail.com

October 1–4

SIGUCCS '17: ACM SIGUCCS Annual Conference, Seattle, WA, Sponsored: ACM/SIG, Contact: Cate Lyon, Email: lyoncd@whitman.edu

October 4–7

SIGITE/RIIT 2017: The 18th Annual Conference on Information Technology Education and the 6th Annual Conference on Research in Information Technology (RIIT), Rochester, NY, Sponsored: ACM/SIG, Contact: Stephen Zilora, Email: sjz@it.rit.edu

October 9–13

MobiCom '17: The 23rd Annual International Conference on Mobile Computing and Networking, Snowbird, UT, Sponsored: ACM/SIG, Contact: Jacobus Van Der Merwe, Email: kobus@cs.utah.edu

October 12–14

SEC '17: IEEE/ACM Symposium on Edge Computing, San Jose/Silicon Valley, CA, Contact: Junshan Zhang, Email: junshan.zhang@asu.edu

October 14–18

MICRO-50: The 50th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, MA, Contact: Hillery C. Hunter, Email: hhunterjaeger@hotmail.com

XRDS

Seeks Student Volunteers

Are you a student who enjoys staying up to date with the latest tech innovations, and is looking to make an impact on the ACM community?

Editorial positions are now open.

For more information and to apply visit:
<http://xrds.acm.org/volunteer.cfm>



Association for
Computing Machinery



minimize market impact. Now, robots match market participants wishing to transact, without third-party intermediation. In so doing they increase liquidity at a much larger scale and with more efficiency.

Lack of open access is not the only way in which a finance platform may be incomplete. Yahoo Finance, for example, is incomplete because it lacks key business processes to complete transactions. Ratings agencies are incomplete because most of their key value-adding business processes require human expertise and are not amenable to codification in IT systems.

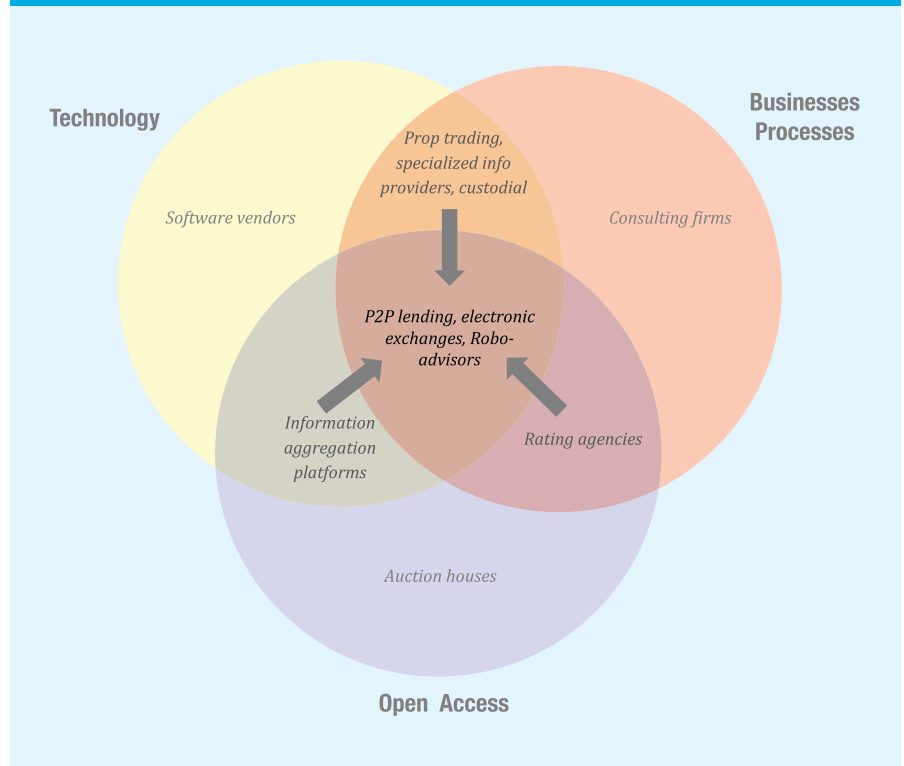
The Venn diagram shown in the accompanying figure provides a graphical view of our framework, which combines the three platform components and also provides some examples. The intersection at the center defines complete platforms; all of the other regions denote incomplete ones. The Venn diagram representation is intuitive. The businesses on the periphery of the diagram tend to be those that play supporting roles, rather than central ones, in the life cycle of financial transactions. The platform framework can be useful for analyzing incumbent and new entrant business models in

a number of ways. To start, the framework identifies which capabilities are required for each type of platform to attain completeness.

Conversely, the framework suggests which business functions are vulnerable to different types of disruptions due to their incompleteness. It also provides a way to think about transitions from partial to complete platforms, the implications of these, and the opportunities that would motivate such transitions. Finally, the framework permits us to examine new technologies in terms of the businesses they are likely to impact. Technologies that facilitate platform completion along a specific dimension will likely affect businesses that are incomplete along that dimension.

Complete platforms may also be created, by design, *de novo*. Amazon and PayPal are early examples in retail and payments. Peer-to-peer lending and robo-advisor platforms are recent instances of complete platforms in finance. Lending Tree, for example, is “open,” and provides the technology infrastructure and processing required to connect lenders and borrowers directly. Robo-advisors are openly accessible to retail investors and aim

The three core components of a complete platform with examples of platform businesses exhibiting various levels of completeness.



to do much of what human advisors have traditionally done—screening investments and providing standard analytics like portfolio optimization—through technology.

For businesses that are currently based on incomplete platforms, the diagram also suggests approaches to platform completion. One strategy for transitioning from a legacy incomplete platform to a complete platform is to add a missing component. This *platform completion* strategy can be pictured schematically by considering arrows originating at peripheral points on the diagram and terminating in the center. An example of this form of platform completion from a non-finance domain is the expansion of Angie's Lists from a home services rating system to home services booking system, through the addition of a process for conducting transactions. In finance, an equivalent example might be if Google or Yahoo Finance were to introduce capabilities like trading, investment advice, asset management (or even investment services) on their platforms.

An alternative strategy for FinTech platform completion is *component replacement*, aimed at introducing functionality that is cheaper, faster, or safer, relative to legacy platform components. Such replacement strategies most often seek to leverage the increasing speed, connectivity, and safety of technology components. A compelling FinTech example is the potential for Blockchain technology to replace legacy post-trade processes that currently require trusted third parties such as clearinghouses and depositories to manage and administer the clearing, settlement, and custody associated with trades and payments. Properly implemented, Blockchain-based systems could significantly reduce the need for specialized custodial institutions since, in principle, market participants would be able to exchange and record transactions securely in the cloud. This new technology, which essentially uses crowdsourcing to implement distributed ledgers, is currently too slow and expensive to replace existing practices. However, the expectation is that speed and cost improvements will displace platform components that address post-transaction processes. Indeed, Blockchain could serve as the core technology for other future com-

The broad implications for FinTech platforms are clear.

plete platforms that require verification and trust.

Emerging AI technologies could be similarly disruptive in replacing existing business processes that involve basic human perception and reasoning about both structured and unstructured data from text, images, and sound. As the volume of unstructured data continues to increase, the artificial intelligence functionality for interpreting and acting on it automatically will likely replace a growing number of human-intensive processes.

Dominant AI platforms are already beginning to emerge. They are fueled by their access to “big data” (searches, purchases, and posts from participants using their technology) and are well positioned to become critical components of complete FinTech platforms of the future, where the platform owners can exploit their expertise in cloud computing, secure transaction processing, search and optimization, without incurring the up-front costs of formally entering the finance industry as full-blown competitors.

In the current regulatory landscape, platform completion and component replacement in FinTech seem likely to occur mostly through platform partnerships, driven by the forces of regulation and economics. Customer acquisition and regulatory compliance activities in finance can be very expensive for newcomers, making it difficult to dislodge incumbents. At the same time, incumbents tend to resist ceding access to their customers, and this aversion will likely induce them to pursue strategies centered on acquiring innovators for platform completion or component replacement while exploiting the existing levels of trust embodied in their brands. Finally—and perhaps most significantly—even though they are not natural

participants today, once market participants begin to trust the data-handling and AI capabilities of the large Internet platforms, they could become core components of FinTech platforms for the same reasons they are trusted in the social, retail, and device domains.

Indeed, partnerships between technology platforms and financial services franchises are already emerging. H&R Block's partnership with IBM's Watson in the tax arena is a recent case in retail finance. Just as H&R Block chose not to build its own version of Watson from scratch, other financial players are unlikely to choose to build Google's or Amazon's artificial intelligence and machine-learning capabilities from scratch.

The broad implications for FinTech platforms are clear. If future investors and consumers of financial services begin to trust FinTech platforms in the ways they have for retail and travel, then financial advisory and intermediation activity may well go the way of manufacturing and brick-and-mortar retail stores. In such a connected world of data-intensive platforms, in which products and services can be micro-tailored to specific clients, and in which trusted platforms provide a broad spectrum of these products and services to these clients, will future generations really care if they buy stock through Fidelity, Google, or Amazon ... or through some combination? **C**

References

1. Dhar, V. When to trust robots with decisions and when not to. *Harvard Business Review* (May 2016); <http://bit.ly/23W1xFi>
2. Dhar, V. and Stein R.M. FinTech platforms and strategy. MIT Sloan Research Paper No. 5183-16, 2017; SSRN link: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2892098
3. Dhar, V. and Sundararajan, A. Information technologies in business: A blueprint for education and research. *Information Systems Research* 18, 2 (June 2007); <http://bit.ly/2waHo8o>
4. Moss, D. and Kintgen, E. The Dojima Rice Market and the Origins of Futures Trading. HBR Case 9-709-044. Harvard Business School, 2010.
5. Parker, G. and Van Alstyne, M. Platform strategy. In *The Palgrave Encyclopedia of Strategic Management*, Macmillan, Hampshire, U.K., 2015.
6. Van Alstyne, M., Parker, G., and Choudary, S. Pipelines, platforms, and the new rules of strategy. *Harvard Business Review* (Apr. 2016); <http://bit.ly/1RfZaZM>

Vasant Dhar (vdhar@stern.nyu.edu) is a professor at the Stern School of Business and the Center for Data Science at New York University.

Roger M. Stein (rms@rogermstein.com) is an adjunct professor of Finance at NYU's Stern School of Business and a research affiliate at the MIT Laboratory for Financial Engineering.

Copyright held by author.



Kode Vicious

IoT: The Internet of Terror

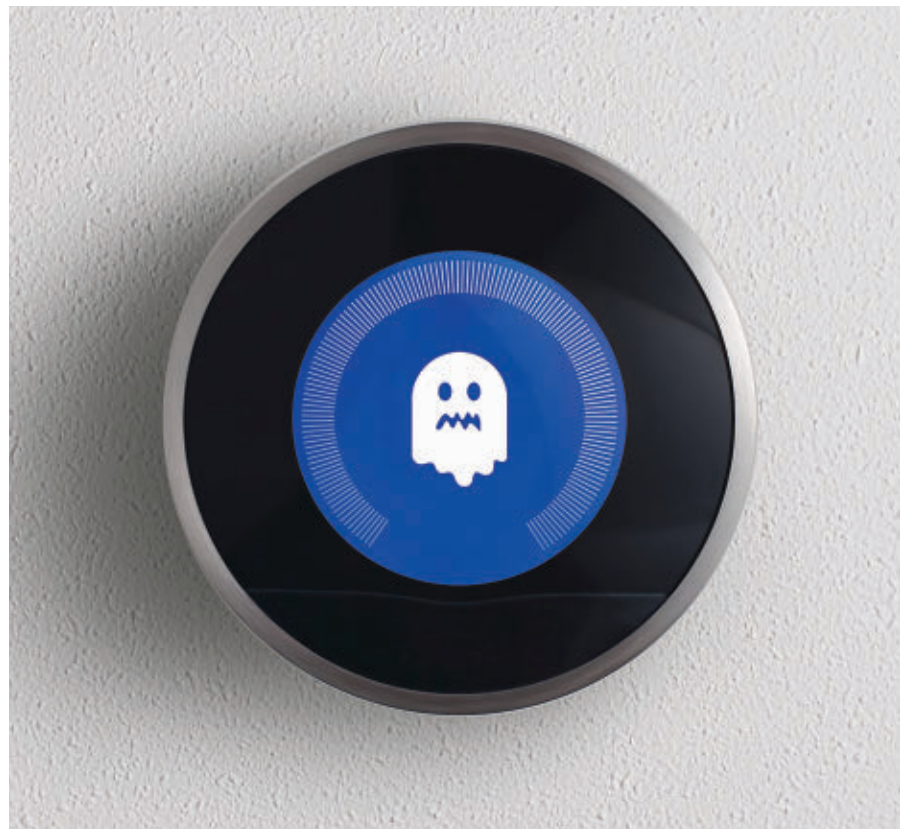
If it seems like the sky is falling, that's because it is.

Dear KV,

We are deploying a consumer IoT (Internet of Things) device, with each device connected to a cloud service that acts as the platform from which it will be controlled. The device itself is not dangerous: it is a simple, slimmed-down tablet to be used in hotel chains to replace an alarm clock and TV remote, and to provide access to room service. The device is battery operated, rechargeable, and cheap enough that hopefully no one will want to steal it. Guests cannot load any information into it, and—unlike a typical tablet—it does not serve as a Web browser.

We have an engineer who seems overly worried about the security of the communication between the device and the back end. We are working on the alpha version of the system now. All communication between the device and the cloud is unencrypted, and no user data crosses the network. Who cares what TV channel you are watching or what you ate for breakfast? On such a lightweight embedded device, the battery-life cost of encryption is significant and reduces battery life by 25% in our tests. We expect many users will not replace them in their charging cradles, since most people cannot remember to charge their own phones every night; thus, battery life will be very important to the product.

Even if we do turn on some encryption, we would like it to be as little as possible, again, to preserve battery life. I know a longer key is harder to break, but it also means that we will be using a



lot more of the battery protecting what is, in reality, data that is hardly a state secret. What do you think is the right level of encryption, if any, to use in such a system, and how can we get the annoying engineer to shut up?

Not So Secret

Dear Not So,

It is true that many security-focused engineers can sound like Chicken Little,

running around announcing that the sky *is* falling, but, unless you have been living under a rock, you will notice that, indeed, the sky is falling. Not a day goes by without a significant attack against networked systems making the news, and the Internet of Terror is leading the charge in taking distributed systems down the road to hell—a road that you wish to pave with your good intentions.

Before I get to the question of encryption and key length, I would like

to point out two things. An IoT device is nothing more than an embedded system with a TCP/IP stack. It is not a magical object that is somehow protected from attackers because of how cool, interesting, or colorful it is.

Second, and I cannot believe I have to point this out to people who would read or write to KV, but the Internet has a lot of stuff on it, and it is getting bigger every day. Once upon a time, there were fewer than 100 nodes on the Internet, and that time is long past. KV has three Internet-enabled devices within arm's reach, and, if you think a billion users of the Internet aren't scary, try multiplying that by 10 once every fridge, microwave, and hotel alarm clock can spew packets into the ether(net).

Let's get this straight: if you attach something to a network—any network—it had better be secured as well as possible, because I am quite tired of being awakened by the sound of the gnashing of teeth caused by each and every new network security breach. The Internet reaches everywhere, and if even one-tenth of 1% of the people on it are bad actors, then that is one million potential attackers across the earth, and each one may be in control of far more than three devices!

Encryption, in and of itself, is not a solution to securing a system. There are many components that go into building a secure system and service, most of which I will not go into here due to limitation of space and my blood pressure medicine, which my doctor tells me not to grind between my teeth. Encrypted communication is one tool to be used when securing a networked service. You say there are no “state secrets” in your system, and, I guess, so long as the president is not staying in one of your IoT-equipped hotels, perhaps that is true. Even if the president no longer eats children for breakfast, there are plenty of other risks involved in building a system as you have described.

If attackers gain access to the private Wi-Fi network the hotel uses to deploy your system—which is likely, because the hotel will probably pick a password such as “1234567890”—then they can see all the traffic used to perform operations. Want to wake your neighbors at 01:00, 03:00, and 05:00? Simple, just create a request

Encryption, in and of itself, is not a solution to securing a system.

with their room number and the appropriate settings. Perhaps you would like to put a hotel out of business? Easy enough, just order expensive champagne from room service for everyone staying in the hotel. Hotel guests will gladly accept it and then sue to have it removed from their bills. Attacks do not have to occur at the NSA/FSB/GCHQ or similar levels to run you and your customers out of business. I think you can see why your engineer is talking about encryption.

You did not mention authentication at all, so let's talk a bit about that as well, because authentication protocols also require the use of cryptographic algorithms—the ones, you complain, that drain your batteries. The system you describe needs the ability to discern who made a request as a way of preventing the attack I just described, where I ordered champagne for everyone in the hotel. How are you going to ensure that room 243 really ordered those drinks?

Now that I have convinced you that connections to the back-end system need to be both encrypted and authenticated, we can split hairs on how big a key to use, and, you know what they say, the bigger the key, the deeper the lock! Choosing algorithms and key sizes is where we can start to talk about device features and power usage.

At the moment, the AES (Advanced Encryption Standard) set of algorithms is what is used most often for encryption, and this protocol has at least three standard key lengths: 128, 192, and 256 bits. There are various recommendations on key lengths, but I rather like the comparison of several sets of recommendations on this website: <https://www.keylength.com/en/4/>. NSA—whoops, I mean NIST—

suggests AES-128 as a minimum from 2016 through 2030, and as your system (likely) does not contain state secrets, it ought to be sufficient for your use.

Many embedded microprocessors now come with dedicated circuits to offload cryptographic operations. The offloading of crypto algorithms is nothing new and is subject to the same “cycle of life” that we see in other areas of software and systems engineering, where specialized services first appear as add-on peripherals, then show up in the CPU itself. In the 1990s, microprocessors were unable to keep up with the then-new algorithms being used to set up virtual private networks (VPNs), so companies produced cryptographic offload chips.

Processors got faster and larger, and what could not be handled by frequency scaling was put into special instructions to handle the harder parts of crypto. Embedded processors, like all processors, continue to gain more circuits, including those for cryptography. Any device, such as yours, that drives a display is bound to have some sort of cryptographic engine, and the only reason to ignore it is sheer laziness. It is very likely that your embedded processor already has acceleration for AES, and if it does not, spend a quarter and buy a better one.

KV

Related articles on queue.acm.org

Kode Vicious Battles On

George Neville-Neil

A koder with attitude, KV answers your questions. Miss Manners he ain't.

<http://queue.acm.org/detail.cfm?id=1059801>

Pervasive, Dynamic Authentication of Physical Items

Mandel Yu and Srinivas Devadas

The use of silicon PUF circuits

<http://queue.acm.org/detail.cfm?id=3047967>

Security Collapse in the HTTPS Market

Axel Arnabak, et al.

Assessing legal and technical solutions to secure HTTPS

<http://queue.acm.org/detail.cfm?id=2673311>

George V. Neville-Neil (kv@acm.org) is the proprietor of Neville-Neil Consulting and co-chair of the ACM Queue editorial board. He works on networking and operating systems code for fun and profit, teaches courses on various programming-related subjects, and encourages your comments, quips, and code snips pertaining to his *Communications* column.

Copyright held by author.

Viewpoints

What Can Agile Methods Bring to High-Integrity Software Development?

Considering the issues and opportunities raised by Agile practices in the development of high-integrity software.

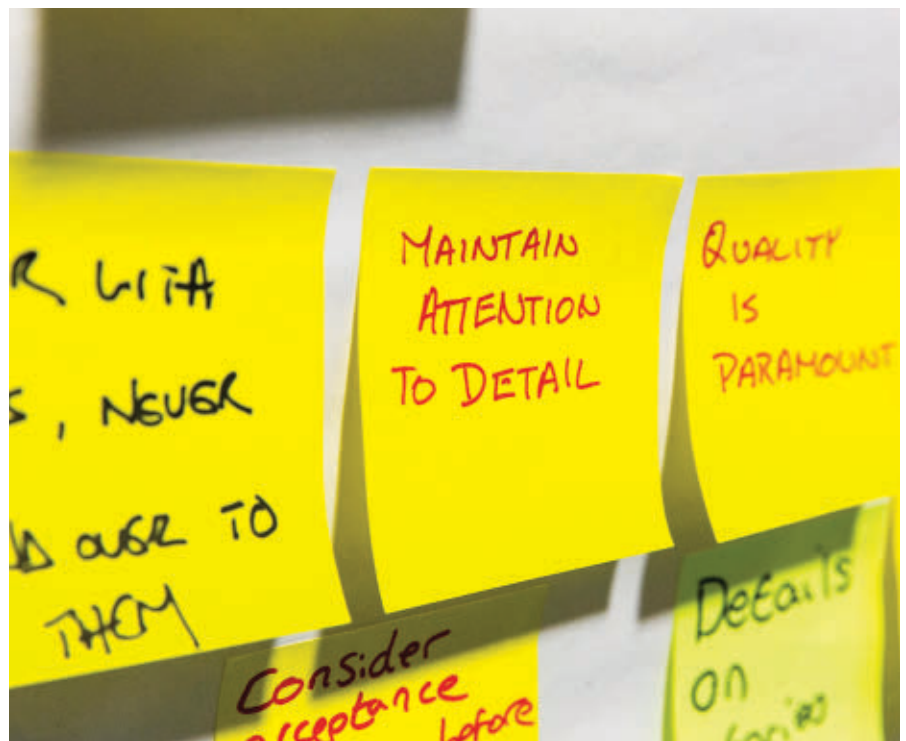
THERE IS MUCH interest in Agile engineering, especially for software development. Agile's proponents promote its flexibility, lean-ness, and ability to manage changing requirements, and deride the plan-driven or waterfall approach. Detrac-tors criticize Agile's free-for-all.

At Altran U.K., we use disciplined and planned engineering, particu-larly when it comes to high-integrity systems that involve safety, security, or other critical properties. A shallow analysis is that Agile is anathema to high-integrity systems development, but this is a naïve reaction. Pertinent questions include:

- ▶ Is Agile compatible with high-integrity systems development?
- ▶ Where is Agile inappropriate?
- ▶ Do Agile's assumptions hold for high-integrity or embedded systems?
- ▶ Could high-integrity best-practice improve Agile?

We don't have all the answers, but we hope this Viewpoint continues to provoke debate on this important topic.

Why bother with Agile at all? We often encounter two myths regarding the "traditional" approach to high-integrity software development: that we somehow manage to perform a single-iteration waterfall style process, and that "formal" notations



are not amenable to change. Neither myth rings true with our experience. As our projects develop, they must absorb change and respond to defects just like any other. This led to an observation: your project is going to become iterative whatever you do, so you might as well plan it that way from the beginning. This lesson was put to good effect in the MULTOS CA project,⁶ which initially planned for

seven iterations, but delivered after 13, owing to a barrage of change requests. Nevertheless, it worked, and we were able to keep a substantial formal specification and all the other design documentation up to date as the project evolved. Knowing that "change" and "iteration" were at the heart of the Agile manifesto, we decided to see what we could learn and bring to future projects.

Background and Sources

Many consider Agile as beginning with XP,¹ but its roots are much older. Many of XP's core practices were well established long ago—their combination and rigorous practice was novel. A survey⁹ notes that both incremental and iterative styles of engineering were used in the 1950s. Redmill's work on evolutionary delivery¹² predicted many of the problems faced by Agile projects. Boehm² provides some useful insight, while the development of MULTOS CA⁶ compared Correctness-by-Construction with XP,³ showing that the two were not such strange bedfellows after all.

Lockheed Martin developed the Mission Computers for the C130J by combining semi-formal specification, strong static verification, iterative development, and a strongly Lean mindset.¹¹ Use of Agile has been reported by Thales Avionics,⁵ while SINTEF have reported success with SafeScrum.¹³ A recent and plain-speaking evaluation of Agile comes from Meyer,¹⁰ although he does not specifically deal with high-integrity issues.

Agile Assumptions and Issues

How do Agile's practices and assumptions match real high-integrity projects? Here are some of the most obvious clashes. For each issue, we start with a brief recap of the practice in question, then go on to describe the issue or perceived clash, followed by our ideas and experiences in overcoming it. Where possible, we close each section with an example of our experience from the C130J, MULTOS, or iFACTS projects.

Dependence on "Test"

Agile calls for continuous integration, with a regression test suit, and a test-first development style, with each function associated with specific tests. Meyer calls these practices "brilliant" in his summary analysis,¹⁰ but Agile assumes that dynamic test is the principal (possibly only) verification activity, saying when refactoring is complete, or when the product is good enough to ship.

The safety-critical community hit the limits of testing long ago. Ultra-reliability cannot be claimed from "lots of testing." Security is even more

How do Agile's practices and assumptions match real high-integrity projects?

difficult—corner-case vulnerabilities, such as HeartBleed—defy an arbitrarily large amount of testing and use. In high-integrity development, we use diverse forms of verification, including checklist-driven reviews, automated static verification, traceability analysis, and structural coverage analysis.

There is no barrier between these verification techniques and Agile, especially with an automated integration pipeline. We try to use verification techniques that complement, not repeat each other. If possible, we advocate for *sound* static analyses (tools that find *all* the bugs, not just some of them), since this gives greater assurance and reduces pre-test defect density. With careful consideration of the assumptions that underpin the static,⁸ we can reduce or entirely remove later testing activities.

The NATS iFACTS system⁴ augments the software tools available to air-traffic controllers in the U.K. It supplies electronic flight-strip management, trajectory prediction, and medium-term conflict detection for the U.K.'s en-route airspace, giving controllers substantially improved ability to plan ahead and predict potential loss-of-separation in a sector. The developers precede commit, build, and testing activities with static analysis using the SPARK toolset. Overnight, the integration server rebuilds an entire proof of the software, populating a persistent cache, accessible to all developers the next morning. Working on an isolated change, the developers can reproduce the proof of the entire system in about 15 minutes on their desktop machines, or in a matter of seconds for a change to a single module. While Agile projects might have a "don't break the tests" mantra, on iFACTS it's "don't

break the proof (or the tests) ..."

Upfront Activities and Architecture. Agile advocates building what is needed now, using refactoring to defer decisions. Refactoring must be cheap, fast, and limit rework to source code.

Our principal weapon in meeting non-functional requirements is *system* (not just software) architecture, including redundancy and separation of critical from non-critical. Such things can be prohibitively expensive to refactor late in the day. We need just enough upfront architecture work to argue satisfaction of key properties. We also do a "What If?" exercise to ensure the proposed architecture can accommodate foreseeable changes.

The MULTOS CA project had some extraordinary security requirements, which were met by a carefully considered combination of physical, operational, and computer-based mechanisms. The software design was much simplified as a result of this whole system view. The physical measures included the provision of a bank vault and enclosing Faraday cage—hardly items that we could have ignored and then "refactored in" later.

User Stories and Non-Functional Requirements. For security and safety, we must ensure our specification covers all possible inputs and states. Agile uses stories to document requirements, but these sample behavior, with no completeness guarantee. The gaps between stories may contain vulnerabilities, bugs, unexpected termination, and undefined behavior. Meyer files user stories under "Bad and Ugly," and we agree.

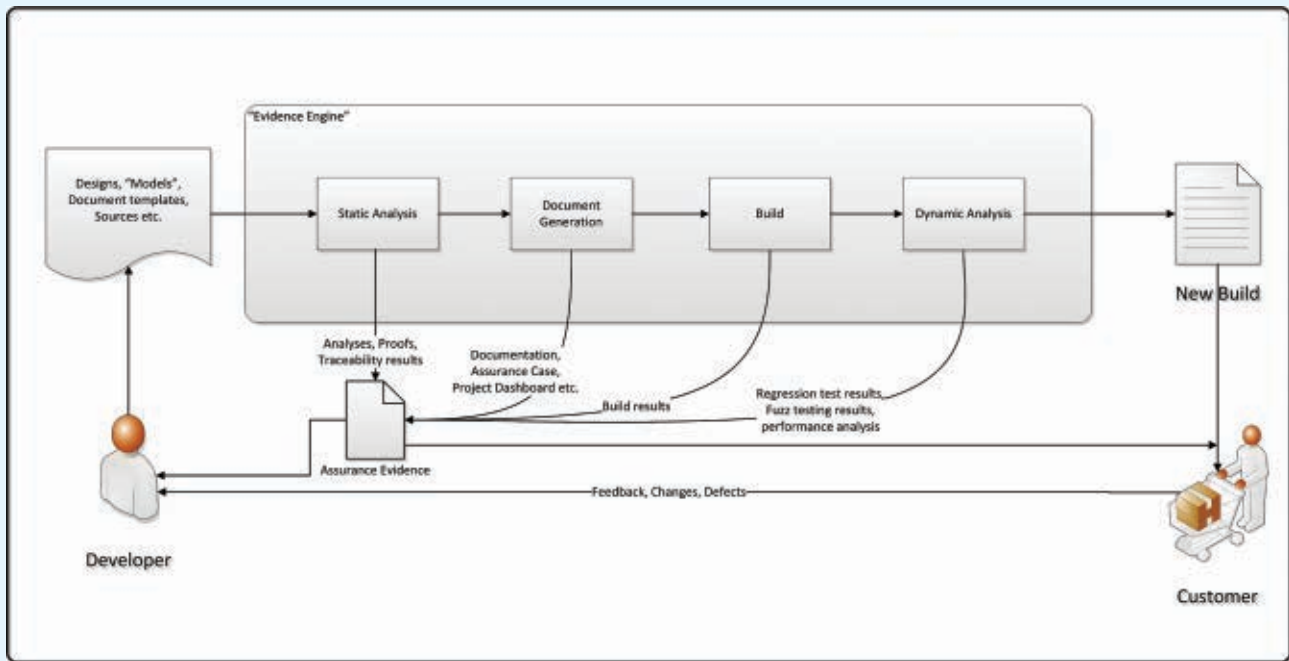
For critical systems, we prefer a (semi-)formal specification that offers some hope of completeness. The C130J used Parnas tables to specify critical functions. They seemed to work well—they were simple enough to be understood by system engineers, yet sufficiently formal to be implemented and analyzed for correctness.

Sprint Pipeline. Agile usually requires a single active "Sprint," delivered immediately to the customer, so only two builds are ever of interest:

- ▶ Build N: in operation with the customer; used to report defects.
- ▶ Build N+1: the current development sprint.

This assumes the customer is al-

High-integrity Agile evidence engine.



ways able to accept delivery of the product and use it immediately. This is not realistic for high-integrity projects. Some customers have their own acceptance process, and regulators may have to assess the system before deployment. These processes can be orders-of-magnitude slower than a typical Agile tempo.

iFACTS uses a deeper pipeline and multiple iteration rates, with at least four builds in the pipeline:

- Build N: in operation with the customer.
- Build N+1: undergoing customer acceptance. This process is subject to regulatory requirements, and so can take months.
- Build N+2: in development and test.
- Build N+3: undergoing requirements and formal specification.

All four pipeline stages run concurrently with multiple internal iteration rates and delivery standards. The development team can deliver to our test team several times a day. A rapid build can be delivered to the customer (in, say, 24 hours), but comes with limitations on its assurance package and allowed use: it is not intended for operational use, but for feedback from the customer on a new feature. A full

build (perhaps once every six months) has a complete assurance package, including a safety case, and is designed for eventual operation. The trick is to make the iteration rates *harmonic*, both with each other and with the customer and regulator's ability to accept and deploy releases.

Embedded Systems Issues. Agile presumes plentiful availability of fast testing resources to drive the development pipeline. For embedded systems, if the hardware exists, there may be just one or two target rigs that are slow, hostile to automation, and difficult to access. We have seen projects revert to 24-hour-a-day shift-working to allow access to the target hardware.

Agile presumes plentiful availability of fast testing resources to drive the development pipeline.

In mitigation, we reduce on-target testing with more static verification. Secondly, if we know that code is completely unambiguous, then we can justify testing on host development machines and reduce the need to repeat the test runs on target. Hardware simulation can give each developer a desktop virtual target or a fast cloud for the deployment pipeline. While virtualization of popular microprocessors is common, high-fidelity simulation of a target's operating environment remains a significant challenge.

On one embedded project, all development of code, static analysis, and testing is done on developers' host machines, which are plentiful, fast, and offer a friendly environment. A final re-run of the test cases is performed on the target hardware with the expectation of pass-first-time, and allowing the collection of structural coverage data at the object-code level.

Opportunities

High-integrity practices can complement Agile. We previously mentioned the use of static verification tools. While we have a preference for developer-led, sound analysis, we recognize that some projects might find more benefit in unsound, but easier to

adopt, technologies, such as bounded model checking. Computing power is readily available to make these analyses tractable at an Agile tempo.

A second opportunity comes with the realization that, if we can automate analysis, building and testing of code, why not automate the production of other artifacts, such as synthesis of code from formal models, traceability analysis, and all the other documentation that might be required by a particular customer, regulator, or standard? An example of such an “Evidence Engine” is shown in the accompanying figure.

Commercial Issues

A crucial issue: How can we adopt an Agile approach, and still estimate, bid, win, and deliver projects at a reasonable profit? Our customers’ default approach is often to require a competitive bid at a fixed price, but how can this be possible in an Agile fashion if we are brave enough to admit that *we don’t know everything* at the start of a project? In most of our projects, the users, procurers, and regulators are distinct groups, all of whom may have wildly different views of what “Agile” means anyway.

We have had good experience with a two-phase approach to contracting, akin to the “architect/builder” model for building a house. Phase 1 consists of the “Upfront” work—requirements, architectural design, and construction of a skeletal satisfaction argument. The “just enough” termination criteria for phase 1 are:

- Convincing evidence that the architecture will work, meet non-functional requirements, and can accommodate foreseeable change.

- An estimate of the size (and therefore cost) of the remaining work, given the currently understood scope.

- Established ground rules for agreeing the scope, size, and additional cost of change requests, and commitment to the tempo of decision making for triage of changes and defects.

Phase 2 (possibly a competitive bid) could be planned as an iterative development, using the ideas set out here. The MULTOS CA was delivered in this fashion, with phase 1 on a time-and-materials basis, and phase 2 on a capped-price risk-sharing contract.

How can we adopt an Agile approach, and still estimate, bid, win, and deliver projects at a reasonable profit?

High-Integrity Deployment Pipeline

We have used the ideas described in this Viewpoint at Altran, but we have yet to deploy all of them at once. An idealized Agile development process would use:

- Principled requirements engineering,⁷ concentrating initially on non-functional requirements and development of architecture, specification, and associated satisfaction arguments.

- A rolling formal specification, with just enough formality to estimate the remaining work and opening development iterations.

- An evidence engine, combining static verification, continuous regression testing, automated generation of documents and assurance evidence, and a cloud of virtualized target platforms for integration and deployment testing.

- A planned, iterative development style, starting with a partial-order over system infrastructure and features that exposes potential for parallel development. Early iterations are planned in detail, while the plans for later iterations are left open to accommodate change.

Conclusion

Returning to the questions posed at the beginning of this Viewpoint, we could summarize our findings as follows:

- *No clash*: continuous integration, verification-driven development style, continuous regression testing, and an explicitly planned iterative approach.

- *Potential clash or inappropriate*: overdependence on test as the sole ver-

ification activity, minimizing upfront activities in the face of non-functional requirements, the incompleteness of user stories (especially for secure systems), the need to align sprints and iteration rates with customers and regulators ability to accept deliveries, and the (non-)availability of test hardware for embedded systems.

- *Agile assumptions*: customer decision-making power and tempo, availability of plentiful test hardware, and commercial and contractual models needed to “procure Agile.”

- *Opportunities*: Adoption of formal languages, automated synthesis, and static verification as part of the deployment pipeline. Generalization of continuous integration into an “Evidence Engine.”

We are deploying these ideas on further projects, and look forward to being able to report the results. We hope others will do the same. 

References

1. Beck, K. *Extreme Programming Explained: Embrace Change*. Addison Wesley, 1999.
2. Boehm, B. and Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed*. Addison Wesley, 2003.
3. Chapman, R. and Arney, P. Static verification and extreme programming. In *Proceedings of the ACM SIGAda Conference* (2003).
4. Chapman, R. and Schanda, F. Are we there yet? 20 years of industrial theorem proving with SPARK. In *Proceedings of Interactive Theorem Proving 2014*. Springer LNCS Vol. 8558, (2014), 17–26.
5. Chenu, E. *Agility and Lean for Avionics*. Thales Avionics, 2009; <http://www.open-do.org/2009/05/07/avionics-agility-and-lean/>
6. Hall, A. and Chapman, R. Correctness by construction: Building a commercial secure system. *IEEE Software* 19, 1 (2002), 18–25.
7. Jackson, M. *Problem Frames*. Pearson, 2000.
8. Kanig, J. et al. Explicit assumptions—A prelude for marrying static and dynamic program verification. In *Proceedings of Tests and Proofs 2014*. Springer-Verlag LNCS, 8570, (2014), 142–157; DOI: 10.1007/978-3-319-09099-3_11
9. Larman, C. and Basili, V. Iterative and incremental development: A brief history. *IEEE Computer*, 2003.
10. Meyer, B. *Agile! The Good, the Hype, and the Ugly*. Springer, 2014.
11. Middleton, P. and Sutton, J. *Lean Software Strategies*. Productivity Press, 2005.
12. Redmill, F. *Software Projects: Evolutionary vs. Big-Bang Delivery*. Wiley, 1997; <http://www.safetycritical.info/library/NFR/>.
13. SINTEF. SafeScrum website, 2015; <http://www.sintef.no/safescrum>.

Roderick Chapman (rod@proteancode.com) is an independent consultant software engineer, and an honorary visiting professor at the University of York, U.K.

Neil White (neil.white@altran.com) is Director of the Intelligent Systems Expertise Centre of Altran U.K.

Jim Woodcock (jim.woodcock@york.ac.uk) is Professor of Software Engineering in the Department of Computer Science at the University of York, U.K.

Thanks to Felix Redmill, Jon Davies, Mike Parsons, Harold Thimbleby, and *Communications*’ reviewers for their comments on earlier versions of this Viewpoint.

Copyright held by authors.

**Code is a story that explains
how to solve a particular problem.**

BY ALVARO VIDELA

Metaphors We Compute By

IN THEIR NOW classic book *Metaphors We Live By*,⁶ George Lakoff and Mark Johnson set out to show the linguistic and philosophical worlds that metaphor isn't just a matter of poetry and rhetorical flourish. They presented how metaphor permeates all areas of our lives, and in particular that metaphor dictates how we understand the world, how we act in it, how we live in it. They showed that our conceptual system is based on metaphors, too, but since we are not normally aware of our own conceptual system, they had to study it via a proxy: language.

By studying language, Lakoff and Johnson tried to understand how metaphors work by imposing meaning in our lives. The basic example they present is the conceptual metaphor “argument is war.” We understand the act of arguing with another person in the same way we understand war. This leads to the following expressions in our daily language:

- ▶ Your claims are *indefensible*.
- ▶ He *attacked every weak point* in my argument.
- ▶ I *demolished* his argument.
- ▶ I never *won* an argument with him.

These sentences may seem innocuous, but the problem is how we act and feel based on them. We end up seeing the person we are arguing with as our *opponent*, someone who is *attacking* our positions, so we structure our arguments as if we were at war with the other person. This means the metaphor is not just language flourish; we live it. Lakoff and Johnson propose the exercise of imagining a culture where arguments are not viewed in terms of war—of winners and losers—but in which language is a dance, where you have to cooperate with a partner in order to achieve a desired goal, reaching conclusions as a team.

The book goes on to analyze the different aspects of language and metaphors and how they affect our concepts and view of the world. The authors give many examples to defend the thesis that our understanding of the world is based on metaphors and that those metaphors are the foundation of behavior.

One of the book's biggest takeaways is that metaphors enable certain ways of thinking, while restricting others, as the argument-as-war example illustrates. This article applies this idea to computer science. How do metaphors shape the way we understand computing and its related problems? What kinds of problems are enabled by the metaphors in use? And, no, monads are not like burritos!⁸

First, the article looks at how metaphors help us understand the relatively young world of computers and how they affect the way we structure code or design algorithms and data structures. We even solve problems based on which metaphors are part of our arsenal, or *toolbox*. “Sometimes our tools do what we tell them to. Other times, we adapt ourselves to our tools’ requirements,” states author Nicholas



Carr in his book *The Shallows*.³ Metaphors are the tools of comprehension.

A Metaphorical Understanding

People understand new concepts by relating them to what they already know. Back in the late 1940s and early 1950s when today's computers came to life, no word for such an invention existed, but people understood them as automatic brains. Actually, the word *computer* existed at that time, but it was used to refer to the person who did calculations for engineers. Think of engineers needing to know the trajectory of a projectile or how the wind would affect an airplane's wing shape; they would throw a couple of formulas and numbers to their *human computers* to get

the answer. Now came these new machines that did the work automatically; they called them *electronic computers*, eventually dropping the *electronic* part of the name. So, our very own discipline was named after a metaphor.

But metaphors also obscure possibilities if you do not understand their limitations. A common problem with new metaphors is the original meaning of the word is used at face value. The word being used to explain a new concept may actually limit understanding of that very concept. In his book *The Information*,⁵ James Gleick gives a fascinating account of the invention of the telegraph. The word *tele-graph* means far-writing. Lo and behold, early telegraphs were strange machines that

tried to write at a distance, using one-to-one mapping of letters of the alphabet into wires, which sounds terribly impractical. Around that time, thanks to Louis Braille, people began to understand that language could be coded in a form different from the way it sounds (or is written). Morse code was the next step in improving the telegraph and in understanding that people don't have to "write at a distance" to have actual long-distance communications.

Thanks to mathematician Claude Shannon and others like him, we managed to escape from the problems of the *telegraph* metaphor and build the whole discipline of information theory beginning in the late 1940s. Shannon's seminal book, *The Math-*

ematical Theory of Communication, put forth the basic elements of communication: information must be encoded first, then sent into a channel to be decoded at the other end, so the destination receives it.⁷

Metaphors and Code

A well-known unattributed quote (often misattributed to Charles Baker) is, “To program is to write to another programmer about our solution to a problem.”² A program is an explanation of how a problem might be solved; it’s a metaphor that stands in place of a person’s understanding. For metaphors to be effective, however, they need to convey meaning using concepts already known to us. The explanatory power of a particular program can be used as a measure of its own elegance.

Consider the following example. Say you could program a computer to command other computers to perform tasks, respecting their arrival order. This description is already difficult to understand. On the other hand, you could describe the solution by describing a *queue server* that assigns *jobs* to *workers* based on a *first come first served* queue discipline.

A queue is a familiar concept from daily life, seen at the supermarket, the bank, airports, and train stations. People know how they work, so for someone reading your code, it might be easier to talk about queues, workers, and jobs than trying to explain this setting without using the *queue* metaphor.

By choosing the right metaphor, your program reaches a level of abstraction that requires the least amount of effort for someone foreign to the problem to understand the solution. Also, solving the problem with queues provides a whole mathematical theory for free. Mathematics is itself a field where problems are tackled only when an appropriately expressive language is available to approach them. With the queue metaphor, you are no longer punching blindly in the dark. Now you can analyze and understand the problem with all the tools provided by queueing theory.

Data Structures as Metaphors

Analyzing data structures helps us to see which would be a better fit for the performance characteristics of a



By choosing the right metaphor, your program reaches a level of abstraction that requires the least amount of effort for someone foreign to the problem to understand the solution.



particular problem, but we often forget that a data structure also contains explanatory power.

The choice of data structures helps convey the meaning. You could store a bunch of elements in an array, but what if you need to ensure the elements are unique? Isn’t that what *sets* are for? The underlying representation of your set can still be backed by a plain array, but now your program expresses its intentions in a clearer way. Whenever other programmers read it, they will understand that the elements in your collection must be unique. It is important to realize that a program is just another succession of bits that a computer needs to process. It is the programmer who gives meaning to those bits, so you have to use the right metaphor on top of them to make your program clearer. As has been said, “no one has seen a program which the machine could not comprehend but which humans did.”²

You must strive to make your program as easy as possible for other programmers to understand. Ease of comprehension should be the standard by which programs are measured. Keep in mind that you can arrange code in many different ways to solve a computing problem, but not all those arrangements will favor human communication and understanding. You must ask yourself: By reading my code, will others understand how I solved this particular problem?

Just as metaphors enable certain ways of understanding and limit others, so do data structures. Earlier we saw the problem with using direct metaphors such as the original telegraph. When it comes to data structures, you can see that a set will reveal that its elements are unique, and it will allow you to test if an element is a member of the set. With a linked list, however, you get the idea of traversing its elements one after the other, without being able to skip them. With an array you get the idea that you can address its elements by index. The same can be seen with queues and stacks, two of the most fundamental data structures taught in any algorithms course. Each can be implemented using an array—the difference is that one returns the elements in FIFO (first in, first out) order, while

the other does so in LIFO (last in, first out) order, respectively.

Even if this looks like an everyday thing for most programmers, the moment you choose to use a stack instead of a queue, you are deciding how to explain your program. The stack is a very good metaphor for the collection of items that a program works with, because it tells a future reader of the program in which order to expect the items to process. You don't even need to read how the stack is implemented, since you can assume you will get the items in LIFO order. This is why types are so important in computer science—not types as in static type checking of programs, but types as the *concepts* used to describe programs: persons, users, stacks, trees, nodes, you name it. Types are the characters that tell the story of your program; without types, you just have operations on streams of bytes.

Cognitive Leaps

The goal is to find the right metaphor that describes and explains a problem. As explained earlier with the queueing theory example, a cognitive leap was needed to go from tasks that have to be processed in a certain order to understanding that this is a queueing problem. Once you manage to make the cognitive leap, all the mathematical tools from queueing theory are at your disposal. Graph theory is filled with examples of mundane tasks that, once converted to a graph problem, have well-known solutions. Whenever you ask Google Maps to get you to your destination, Google translates your problem to a graph representation and suggests one or more *paths* in the graph. Graphs are the right metaphor, understood by mathematicians and computers alike. Are there other instances of problems that seem difficult but that can be solved by finding the right metaphor? The distributed-systems literature has a very interesting one.

In the late 1980s Alan Demers and his colleagues from Xerox tried to find a solution to database replication in unreliable networks. They classified their algorithms as “randomized,” explaining them by using the rumor-mongering metaphor: a computer would tell two other computers about an update, then each in turn would

tell two other computers about the update, and so on until the information was replicated across the system. This metaphor gave way to a new area of study called gossip algorithms. The *gossip* metaphor makes the idea easy to explain, but the Xerox team was still lacking the mathematical tools that would help analyze the effectiveness of the algorithms.

During their research, they discovered another metaphor related to their problem: epidemics. They understood their algorithms replicated data the same way in which an epidemic disseminates across a population. By using this new metaphor, they got immediate access to all the knowledge in *The Mathematical Theory of Epidemics*,¹ which fit their work like a glove. Not only did they name their paper “Epidemic Algorithms for Replicated Database Maintenance,”⁴ they also took the nomenclature of that discipline to explain their algorithms. It was a matter of finding the right metaphor to get access to a new world of explanatory power.

Metaphors Everywhere

Do we really use that many metaphors in programming? Let's take a look at an example from the distributed systems literature (metaphors are in *italics*):

Whenever nodes need to *agree* on a common value, a *consensus* algorithm is used to *decide* on a value. There's usually a *leader* process that takes care of making the final decision based on the *votes* it has received from its *peers*. Nodes communicate by sending *messages* over a *channel*, which might become *congested* because of too much *traffic*. This could create an information *bottleneck*, with *queues* at each end of the *channels* backing up. These *bottlenecks* might render one or more nodes *unresponsive*, causing *network partitions*. Is the process that's taking too long to *respond dead*? Why didn't it acknowledge the *heartbeat* and *trigger* a timeout? This could go on, but you get the point.

A Story in Code

Programmers must be able to tell a story with their code, explaining how they solved a particular problem. Like writers, programmers must know their metaphors. Many metaphors

will be able to explain a concept, but you must have enough skill to choose the right one that's able to convey your ideas to future programmers who will read the code.

Thus, you cannot use every metaphor you know. You must master the art of metaphor selection, of meaning amplification. You must know when to add and when to subtract. You will learn to revise and rewrite code as a writer does. Once there is nothing else to add or remove, you have finished your work. The problem you started with is now the solution. Is that the meaning you intended to convey in the first place?

Acknowledgments

Thanks to Jordan West and Carlos Baquero for our discussions about how metaphors permeate computing, and for their feedback for this article. 

Related articles on queue.acm.org

First, Do No Harm: A Hippocratic Oath for Software Developers

Phillip A. Laplante

<http://queue.acm.org/detail.cfm?id=1016991>

Coding for the Code

Friedrich Steimann and Thomas Kühne

<http://queue.acm.org/detail.cfm?id=1113336>

A Nice Piece of Code

George V. Neville-Neil

<http://queue.acm.org/detail.cfm?id=2246038>

References

1. Bailey, N.T.J. *The Mathematical Theory of Epidemics*. C. Griffin and Co., 1957.
2. Baker, C. (Ed.). What a programmer does. *Datamation* (Apr. 1967); http://archive.computerhistory.org/resources/text/Knuth_Don_X4100/PDF_index/k-9-pdf/k-9-u2769-1-Baker-What-Programmer-Does.pdf.
3. Carr, N. *The Shallows*. W.W. Norton, 2011.
4. Demers, A., Greene, D., Hauser, C., Irish, W. and Larson, J. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, (1987), 1–12.
5. Gleick, J. *The Information: A History, a Theory, a Flood*. Pantheon, 2011.
6. Lakoff, G. and Mark J. *Metaphors We Live By*. University of Chicago Press, 1980.
7. Shannon, C.E. and Weaver, W. *The Mathematical Theory of Communication*. University of Illinois Press, 1949.
8. Yorgey, B. Abstraction, intuition, and the 'monad tutorial fallacy'. <https://byorgey.wordpress.com/2009/01/12/abstraction-intuition-and-the-monad-tutorial-fallacy/>

Alvaro Videla (alvaro-videla.com; @old_sound) works as Lead Architect for a major Swiss company. Previously, he was a senior software engineer at Apple and a core developer of RabbitMQ. He is the author of *RabbitMQ in Action*.

Copyright held by owner/author.
Publication rights licensed to ACM. \$15.00.

**Expert-curated guides to
the best of CS research.**

Research for Practice: Technology for Underserved Communities; Personal Fabrication

THIS INSTALLMENT OF Research for Practice provides curated reading guides to technology for underserved communities and to new developments in personal fabrication. First, **Tawanna Dillahunt** describes design considerations and technology for underserved and impoverished communities. Designing for the more than 1.6 billion impoverished individuals worldwide requires special consideration of community needs, constraints, and context. Her selections span protocols for poor-quality communication networks, community-driven content generation, and resource and public service discovery. Second, **Stefanie Mueller** and **Patrick Baudisch** provide an overview of recent advances in personal fabrication (for example, 3D printers).

Their selection covers new techniques for fabricating (and emulating) complex materials (for example, by manipulating the internal structure of an object), for more easily specifying object shape and behavior, and for human-in-the-loop rapid prototyping. Combined, these two guides provide a fascinating deep dive into some of the latest human-centric computer science research results.

As always, our goal in this article is to allow our readers to become experts in the latest topics in computer science research in a weekend afternoon's worth of reading. To facilitate this process, we have provided open access to the ACM Digital Library for the relevant citations from these selections so you can enjoy these research results in full. Please enjoy!

—*Peter Bailis*

Peter Bailis is an assistant professor of computer science at Stanford University. His research in the Future Data Systems group (futuredata.stanford.edu) focuses on the design and implementation of next-generation data-intensive systems.



Technology For Underserved Communities

By Tawanna Dillahunt

According to the Global Multidimensional Poverty Index 2013, 1.6 billion people—or more than 30% of the combined populations of the 104 countries analyzed—were impoverished in terms of health, education, and living conditions.¹ Here, these individuals and those facing similar conditions are referred to as *underserved*.

Designing and building technology to support people in these underserved communities has several complexities. Overcoming these complexities requires the following:

- ▶ Understanding the needs of a specific underserved population and empowering or enabling individuals from that population to produce information and develop their own solutions.
- ▶ Understanding the context and constraints that underserved indi-

viduals often face, such as limited or nonexistent reading skills, digital literacy, Internet and technology access, and, in many cases, infrastructure (for example, no electricity or water). Individuals from underserved populations often face social barriers such as limited social networks, social isolation, and systemic issues that exist beyond our control such as social or income inequality. In addition, these issues and limitations vary from region to region, and platforms that might appear successful in one situation may not apply in another situation.

► Developing technologies that are sustainable or have the ability to maintain themselves in limited-resource environments.

► Considering that technology alone may not meet the needs of certain underserved communities, given the unique factors these communities face. The benefits of technology are often limited to populations that are relatively affluent, educated, and socially connected.

The recently published papers presented here are broadly centered on technology and aim to support the needs of underserved communities. The selection represents a diversity of ACM venues, including the Symposium on Computing and Development, the Conference on Human Factors in Computing Systems, and Designing for Interactive Systems. The selection also represents a diversity of regions—including research from the Global South and Global North—and a diversity of technological artifacts, including one designed to optimize mobile application communication for challenged network environments and both long- and short-term application deployments. The selection also includes efforts and approaches to reach the needs of traditionally underserved groups (for example, low-resource, blind, and unemployed people).

Finally, at least one of the selected works exemplifies technological solutions that: (1) address issues resulting

from poor infrastructures such as unstable network connectivity and electricity (Brunette et al.); (2) enable individuals to produce information, not simply consume information, as suggested by Dell and Kumar² (Vashistha et al.); and (3) support the connection among individuals within a community, or support the connection between low-resource communities and high-resource communities, for information and resource sharing, as suggested by Heeks and Bhatnagar³ (Vashistha et al. and Dillahunt et al.).

Technology Designed to Address Poor Network Environments

Brunette, W., Vigil, M., Pervaiz, F., Levari, S., Borriello, G. and Anderson, R. Optimizing mobile application communication for challenged network environments. In *Proceedings of the 6th Annual ACM Symposium on Computing for Development*, 2015, 167–175; <http://dl.acm.org/citation.cfm?id=2830644>

This paper describes a configurable data-transmission framework designed to minimize network connectivity issues for mobile applications in special contexts. For example, the Open Data Kit (ODK) is a free and open source set of tools that helps organizations in low-resource areas write, field, and manage mobile data collection solutions. ODK is used in mobile-application deployments to collect health and socioeconomic surveys using global positioning system locations and images. Quickly notifying others of important health issues such as an Ebola or polio outbreak in such contexts necessitates reliable data transfer despite challenging network environments.

The authors of this paper contribute ODK Submit, an Android service that coordinates data communication by providing channel-monitoring and transmission-scheduling mechanisms to Android apps. ODK Submit improves deployments by enabling application-level optimization of sparse heterogeneous networks, sends appropriate data over available channels, and identifies available connectivity in challenging network envi-

ronments. It also enables deployment architects (that is, nondevelopers who customize off-the-shelf software packages) to customize mobile-application software to improve data transfer reliability in such challenging environments. This paper contributes a framework for optimizing mobile-application performance for limited-resource settings.

Technology Supporting the Connection Among Individuals within Communities by Enabling the Production of Information

Vashistha, A., Cutrell, E., Borriello, G. and Thies, W. Sangeet Swara: A community-moderated voice forum in rural India. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, 417–426; <http://dl.acm.org/citation.cfm?id=2702191>

Sangeet Swara is a social media Hindi-based voice forum for songs and cultural content that was built using an IVR (interactive voice response) system. Vashistha et al. wanted to understand how such a system might become financially sustainable. To achieve financial sustainability for Sangeet Swara, the authors worked to reduce the cost of a community moderator by having the community moderate the content.

Those calling into Sangeet Swara recorded voice messages, listened to recorded messages, voted on them, and shared them without access to the Internet. Callers helped curate the content by voting on the voice forums as a way of prioritizing playback for others and at the same time eliminating the need for a moderator.

The use of the system was successful, with significant adoption by low-income communities as well as low-income individuals who were visually impaired; the system received more than 25,000 calls and 5,000 voice messages from more than 1,500 people in an 11-week deployment. It did not turn out to be financially sustainable after the toll-free number was replaced with a paid number; however, the opportunity remains to address this issue

going forward.

In summary, the authors contributed Sangeet Swara, a system that empowered low-income community members to produce information and build social capital, which is vital for connecting and strengthening communities. The community moderated the forums by categorizing and rating posts, thus eliminating the need for a single moderator—a step toward creating a self-sustaining system.

Technology Connecting Low-Resource Communities to High-Resource Communities for Information Sharing

Dillahun, T.R., Bose, N., Diwan, S., and Chen-Phang, A.

Designing for disadvantaged job seekers: Insights from early investigations. In *Proceedings of the ACM Conference on Designing Interactive Systems*, 2016, 905–910; <http://dl.acm.org/citation.cfm?id=2901865>

This article focuses on unemployed job seekers from low-income communities in the U.S. The authors conducted a user-centered design process to create and deploy Review-Me, a Web-based application that sourced résumé feedback for individuals who were unemployed from local individuals who were employed. While the application deployment successfully connected job seekers who were currently students, it uncovered limitations and constraints among underserved job seekers. For example, underserved job seekers did not always have access to digital résumés or understand how to recreate physical résumés in digital form. Job seekers rarely had access to an email address, and those with email access often forgot their passwords. Finally, low literacy levels made it very difficult for these job seekers to sign up to use the application.

The results of this work suggested three fundamental design principles to address these shortcomings: compatibility (for example, systems should accept photographed images of résumés); practicality (for example, systems should provide ways to submit résumés offline, such as through kiosks or offline networking devices); and familiarity (for example, systems should allow registration via SMS or familiar mobile accounts such as Instagram or Facebook).

The research findings raise con-

cerns as employers increasingly require job seekers to apply online; government services, such as unemployment benefits, must be applied for online (in some cases, this can be done via phone); and other services, such as health care and even housing, are researched and obtained online. Connecting underserved job seekers with individuals who are employed is commendable. Without making these technologies more inclusive, however, many individuals will continue to be left behind.

Conclusion

For a computer scientist who cares about designing, implementing, and deploying inclusive tools, it is vital to build software and software techniques that aim to:

- Improve poor infrastructures such as unstable network connectivity and electricity or provide solutions that do not require these infrastructures to be stable.

- Empower individuals to produce information, not only to consume it.²

- Connect individuals to others with more resources and help support development within and across communities.

When creating software, computer scientists and practitioners should consider issues that persist in limited-resource areas, such as unstable access to electricity and water and overall poor infrastructure. Designing for other constraints such as limited reading and digital literacy and limited Internet access is often beneficial to populations beyond underserved groups. Empowering individuals to produce information and strengthen bonds within and across communities is critical to reducing systemic issues that exist beyond our control, such as social or income inequality.

References

1. Alkire, S., Roche, J.M. and Seth, S. *Multidimensional Poverty Index 2013*. Oxford Poverty and Human Development Initiative; <http://www.ophi.org.uk/wp-content/uploads/Global-Multidimensional-Poverty-Index-2013-8-pager.pdf?0a8fd7>.
2. Dell, N. and Kumar, N. The ins and outs of HCI for development. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2016, 2220–2232.
3. Heeks, R. and Bhatnagar, S.C. Understanding success and failure in Information Age reform. *Reinventing Government in the Information Age: International Practice in IT-enabled Public-sector Reform*. R. Heeks, (Ed). Routledge, London, 1999, 49–74.

Tawanna Dillahun is an assistant professor at the

University of Michigan's School of Information and holds a courtesy appointment with the electrical engineering and computer science department. She leads the Social Innovations Group (www.socialinnovations.us), an interdisciplinary group specializing in the R&D of ubiquitous and social computing technologies.



Personal Fabrication

By Stephanie Mueller and Patrick Baudisch

Personal fabrication tools such as 3D printers are paving the way to a future in which nontechnical users will be able to create custom objects. With the recent decrease in price for 3D-printing hardware, these tools are about to enter the mass market. The cost of the average consumer 3D printer has dropped from about \$14,000 in 2007 to \$500 today. Given the decreasing price, it is not surprising that the number of consumer 3D printers sold has doubled every year—from 6,000 in 2010 to more than 270,000 in 2015.

While the hardware is now more affordable and the number of people who own a 3D printer is increasing, only a few people use the printers to create 3D models. Most users download models from a platform such as Thingiverse, and after downloading, fabricate them on their 3D printers. At most, users adjust a few parameters of the model, such as changing its color or browsing among predetermined shape options.

Personal fabrication has the potential for more: instead of only consuming existing content, nontechnical users in the future could use 3D printers to create objects that only trained experts can create today. In the past few years, human-computer interaction (HCI) and graphics researchers have worked on a number of challenges to move us toward such a future.

In our paper, “Personal Fabrication” (*Foundations and Trends in Human-Computer Interaction* 10, 3–4, 2017), we provide a comprehensive overview of these challenges and include a survey of more than 200 recent papers. Here, we summarize three of those papers, which are representative of the larger field.

Multimaterial Fabrication

Vidimce, K., Kaspar, A., Wang, Y., Matusik, W. Foundry: Hierarchical material design for multi-material fabrication. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, 2016, 563–574; <http://dl.acm.org/citation.cfm?id=2984516>.

With 3D printing, users can design every aspect of an object: they can create a specific *appearance* (for example, a desired shape, color, and reflectance), a specific *feel* (for example, by printing tactile textures or by using soft materials), and they can make an object perform a desired *function* (for example, using conductive materials for printed electronics or optical clear materials for printing light pipes).

Functional properties can also be achieved by designing the *internal structure* of an object—for example, making an object stand by redistributing its infill to shift its center of mass. Finally, by using microstructures with structurally varying cells, researchers have shown how to emulate different material behaviors using a single material (so-called *metamaterials*).

This paper provides an editing environment for designing such multimaterial composite objects. Composing a set of operators into an operator graph creates the material definitions. The operators are implemented using a domain-specific language for multimaterial fabrication, and users can easily extend the library by writing their own operators.

Domain Knowledge

Umetani, N., Koyama, Y., Schmidt, R. and Igarashi, T. Pteromys: Interactive design and optimization of free-formed free-flight model airplanes. *ACM Transactions on Graphics* 33, 4 (2014), Article 65; <http://dl.acm.org/citation.cfm?id=2601129>.

Professional CAD tools require years of engineering training to gain the necessary expertise as they provide fine control over every parameter in the design. HCI and graphics researchers have looked at how to create design tools that abstract away the necessary domain knowledge by letting users specify the *shape* and *motion* of the desired object; the system then simulates the mechanical be-

havior and either critiques the user's design or automatically adjusts the design to make it comply with respect to *forces*.

Consider even the simple example of designing a paper glider as presented in this paper. Depending on the orientation of the glider, it will be subject to drag forces that make the glider resist the airflow and lift forces that move the glider upward. All forces depend not only on the shape of the glider, but also on its velocity and orientation at a certain moment in time; thus, they are constantly changing as the glider moves through the air. This creates a large parameter space that is infeasible for the user to tackle manually. The design tool described in this paper lets users design the shape of the glider, and as they design, provides real-time feedback on the flight performance.

From Batch-Processing, To Turn-Taking, To Direct Manipulation For Physical 'Data'

Willis, K.D.D., Xu, C., Wu, K.-J., Levin, G., Gross, M.D. Interactive fabrication: New interfaces for digital fabrication. In *Proceedings of the 5th International Conference on Tangible, Embedded, and Embodied Interaction*, 2010, 69–72; <http://dl.acm.org/citation.cfm?id=1935716>.

When simulation is not possible, such as when testing the aesthetics and ergonomics of a design, users have to fabricate the object to evaluate it. Since 3D printing is so slow, users have to think carefully before printing, as every mistake may imply another hour-long or even overnight print.

Planning every step, however, is not feasible for nontechnical users as they lack the experience to reason about the consequences of their design decisions. To solve the problem, researchers proposed repeating the evolution of the user interface from personal computing for personal fabrication. Computing also started with machines that ran a program in one go overnight. Then turn-taking systems such as the command line evolved, which provided users with feedback after every input; finally, direct-manipulation interfaces (such as today's multitouch devices) provided users with continuous feedback during editing. Applying the same interaction concept to fabrication results

in a workflow that looks quite different from what is seen today.

In the new “interactive fabrication” workflow presented in this paper, users work hands-on on the physical workpiece using physical tools (much like in crafting), and see the *physical* workpiece change immediately as they edit. The fast feedback loop allows users to evaluate every intermediate step, allowing them to adjust their decisions along the way. This potentially makes editing of physical data as easy as editing digital data on a multitouch device today.

Conclusion

When will personal fabrication reach consumers? The journey has only just begun. If the starting point is deemed to be 2009 (that is, when the first patents ran out and the first low-cost 3D printer, the MakerBot Cupcake CNC, appeared on the market), then we are clearly still at the very beginning of putting personal fabrication into the hands of consumers.

If personal fabrication today feels like a niche technology for hobbyists, it is most likely because there are still decades to go. We should look at the in-between progress with patience. The success of other technologies such as personal computing could certainly not have been predicted until decades after their inception (consider the words of Thomas Watson, president of IBM, in 1943: “I think there is a world market for maybe five computers”).

If personal fabrication should turn out anything like personal computing in its adoption, we still have an amazing journey ahead of us.

Stefanie Mueller is an assistant professor in MIT's electrical engineering and computer science department, joint with mechanical engineering, and is a member of MIT CSAIL. She develops novel hardware and software systems that advance personal fabrication technologies.

Patrick Baudisch is a professor of computer science at Hasso Plattner Institute at Potsdam University and chair of the Human Computer Interaction Lab. Previously, he worked as a research scientist in the Adaptive Systems and Interaction Research Group at Microsoft Research and at Xerox PARC.

Copyright held by owners/authors.
Publication rights licensed to ACM.

Why the Bell curve hasn't transformed into a hockey stick.

BY THOMAS A. LIMONCELLI

Four Ways to Make CS and IT More Immersive

I was lucky. I learned IT in an incredibly immersive way. My first two jobs were in organizations that followed the very best practices for their day. Because it was all I knew, I considered that to be normal. I had no idea how unique those organizations were. I didn't know at the time that the rest of the industry would not adopt these techniques for a decade or more.

My next career moves brought me in contact with organizations that did not adhere to the same best practices, nor any others. In fact, they were unaware that such best practices existed at all. I considered this to be a bug and went about fixing it, dismayed that anyone would settle for anything else. I was re-creating what I considered “normal.”

The environment I was trying to reproduce, however, was not normal, or more accurately, it was not typical. A typical IT organization is, in comparison, in utter disarray. The quality of IT organizations follows a bell curve: A few percent run like fine-tuned machines, a few percent look like toxic waste dumps on fire, and the vast majority are somewhere in the middle.

Fortunately for me, I won the IT career lottery. Early in my career I saw what the best in class looked like and considered it normal. Later, this high standard made me look like a visionary. The truth is I just didn't know any other way.

Most IT practitioners are not so fortunate. They are not blessed with the same experience I was afforded, and they literally do not know any better.

This, I believe, is why the bell curve has not transformed into a hockey stick, or is even a lopsided blob. This is why we cannot have nice things.

How Did We Get Into This Situation?

Students certainly are not learning best practices in the classroom. In fact, students are more likely to learn the best-of-breed DevOps practices through extracurricular involvement in open-source projects than from their university professors.

Most large open source projects use Git for source-code control, use Jenkins for CI/CD (continuous integration/continuous deployment), and have a fully automated testing procedure because it enables them to scale to large numbers of participants with minimal overhead. Smaller open source projects tend to use these tools, too, because they lack resources and using these tools makes managing the project significantly easier.

Yet, how many universities require CS homework to be turned in via Git commit? How many universities have an IT department that is a showcase for best DevOps practices? How many universities have CS departments and IT departments that collaborate to push the boundaries of best practices? I assure you the number is very low. It is no



surprise that the innovations that led to the DevOps transformation did not come from academia. This is a disgrace that should make every CS department bow their heads in shame.

How Can We Turn This Around?

How can we ensure students are exposed to the best of the best practices from the start so that they consider anything else a bug?

How can we make curricula more immersive?

Here are a few small and big things that universities could do.

1. Use DevOps tools from the start. Students should use source-code repositories such as Git, and CI/CD tools such as Jenkins, as they do their CS homework. These processes should be established as the normal way to work. Professors should expect homework assignments to be turned in by linking to

a Git commit and a Jenkins output log.

Some instructors will undoubtedly feel it is difficult enough to teach first-year computer science without adding the complexities of Git. Most IDEs, however, make simple check-in/check-out operations a breeze, especially for single-person projects with no branches. By the time projects get more collaborative, the students will be ready for the more advanced Git features.

2. Homework should generate a Web page, not text to the console. I recently spoke to a roomful of third-year CS majors and was shocked to learn that most didn't know HTML. The curriculum was fairly standard—undergraduate algorithms and such. HTML was something you learned in the art department; the computer science department was for serious students.

I think there is a middle ground between serious computer science theory

and accidentally turning into a Web application boot camp.

It isn't a radical statement to say that most software engineers write code that is somehow part of a Web-based application. At companies like Squarespace and Google, software engineers' IDEs supply default templates for new programs. Such a template is for a self-contained Web server that directs output to a Web page. Even a simple "Hello World!" program is a Web server that outputs the greeting as a result of an HTTP request and, by default, generates logging information, monitoring metrics, and so on.

Yes, that is a bit much for an introductory student's "Print your name 10 times" program. But after that, generate a Web page!

3. IT curricula should be immersive. How could formal education better emulate the immersive experience that

I was lucky enough to benefit from?

Most IT curricula are bottom up. Students are taught individual subsystems, followed by higher levels of abstractions. At the end, they learn how it all fits together. Toward the end of their college careers, they learn the best practices that make all of it sustainable. Or, more typically, those sustainability practices are not learned until later, when the new graduate has a job and is assigned to a coworker who explains “how things work in the real world.”

Instead, an IT curriculum should start with a working system that follows all the best practices. Students should see this as the norm. They can dissect the individual subsystems and put them back together, rather than building them from scratch.

The Masters of System Administration curriculum at the University of Oslo includes a multiweek immersive experience called the Uptime Challenge.¹ Students are divided into two teams, and each team is given a Web-based application, including multiple Web servers, a load balancer, a database, and so on. The application is a simple social network application called BookFace.

Once the system is running, the instructor enables a system that sends an ever-increasing amount of simulated traffic to the application. Each team's system is checked for uptime every five minutes. The team receives a certain amount of fake money (points) if the site is up, and a small bonus if the page loads within 0.5 seconds. If the site is down, money is deducted from the team. This simulates a typical website business model: you make money only if the site is up. Faster sites are more appealing and profitable. Customers react to down or slow sites by switching to competitors; thus, those lower-performing sites lose money.

The challenge lasts multiple weeks, during which the students learn to perform common web-operation tasks such as software upgrades, bug fixing, task automation, performance tuning, and so on. Inspired by Netflix's Chaos Monkey,² individual hosts are randomly rebooted to test the resiliency of the overall system.

The Uptime Challenge enables students to understand IT's value to the organization and to identify the IT processes that impact this value and permit

continuous improvement. As a result, students are more motivated and better able to assess their own work. This leads to improved engagement and fosters more practical class discussions. It creates a direct feedback loop between a student's actions and the value they create. Most importantly, it better prepares students for the real world.

4. Be immersive from the start. IT projects usually involve some kind of legacy system. The most apt analogy is being asked to change the tires on a truck while it is being driven down the highway.

Software engineers spend more time reading other people's code than writing their own. We evolve existing systems. Green-field or “fresh start” opportunities are rare. Many people I have met have never been in a situation where they designed a new network, application, or infrastructure from scratch. Why can't education better prepare students for this?

Could something like the Uptime Challenge be introduced even earlier in the educational process?

Perhaps on the first day of class students should be handed not only copies of the syllabus, but also the username and passwords to the administrative control panel of a working system. Instruction and labs could be oriented around maintaining this system. Students would have their own wikis to maintain documentation and operational runbooks.

Each student would have their own working system, but I suggest that every few weeks students be randomly reassigned to administer a different system. Seeing how their fellow students had done things differently would be educational. Also, the best way to learn the value of a well-written runbook is to inherit someone else's badly maintained runbook.

Institutions are developing more immersive educational strategies. In cooperation with industry, Bossier Parish Community College^a has created an Associate of Applied Science in Systems Administrator degree, which is highly immersive and covers core DevOps principals, including automa-

a <https://www.bpsc.edu/catalog/current/technologyengineeringmathematics/aas-system-administration.html>

tion technologies (infrastructure as code and software-defined networking), Lean/KanBan ideas, cloud fundamentals, and even DevSecOps.

Conclusion

Education should seek to normalize best practices from the start. Working outside these best practices should be considered a bug. Students should not struggle to learn best practices after graduation, and they should be shocked if potential new employers do not already have these practices in place.

Both IT and CS curricula could be structured to be more immersive, as immersive education more reliably reflects the real world. It prepares students for industry and better informs the research of those who choose that path. Seeing the forest, and then understanding the trees, helps students understand why they are learning something before they learn it. It is more hands-on and therefore more engaging, and lends itself to gamification.

Our first experiences cement what becomes normal for us. Students should start off seeing a well-run system, dissect it, learn its parts, and progressively dig down into the details. Don't let them see what a badly run system looks like until they have experienced one that is well run. A badly run system should then disgust them. ■

Related articles on queue.acm.org

Undergraduate Software Engineering

Michael J. Lutz, et al.

<http://queue.acm.org/detail.cfm?id=2653382>

A Conversation with Alan Kay

<http://queue.acm.org/detail.cfm?id=1039523>

Evolution of the Product Manager

Ellen Chisa

<http://queue.acm.org/detail.cfm?id=2683579>

References

1. Begnum, K. and Anderssen, S.S. The Uptime challenge: A learning environment for value-driven operations through gamification. *Usenix J. Education in System Administration* 2, 1 (2016); <https://www.usenix.org/jesa/0201/begnum>.
2. Tseitlin, A. The antifragile organization. *Commun. ACM* 56, 8 (Aug. 2013), 40–44.

Thomas A. Limoncelli is a site reliability engineer at Stack Overflow Inc. in NYC. He blogs at EverythingSysadmin.com and tweets at @YesThatTom

Copyright held by owner/author.
Publication rights licensed to ACM. \$15.00

Call for Nominations

The ACM Doctoral Dissertation Competition

Background of the Competition

ACM established the Doctoral Dissertation Award program to recognize and encourage superior research and writing by doctoral candidates in computer science and engineering. These awards are presented annually at the ACM Awards Banquet.

Submissions

Nominations are limited to one per university or college, from any country, unless more than 10 Ph.D.'s are granted in one year, in which case two may be nominated.

Sponsorship

Each nomination shall be forwarded by the thesis advisor and must include the endorsement of the department head. A one-page summary of the significance of the dissertation written by the advisor must accompany the transmittal.

Deadline

Submissions must be received by **October 31, 2017** to qualify for consideration.

For Eligibility and Submission Procedure see http://awards.acm.org/doctoral_dissertation/

Publication Rights

Each nomination must be accompanied by an assignment to ACM by the author of exclusive publication rights. (Copyright reverts to author if not selected for publication.)

Publication

Winning dissertations will be published by ACM in the ACM Books Program and appear in the ACM Digital Library. Honorable mention dissertations will appear in the ACM Digital Library.

Selection Procedure

Dissertations will be reviewed for technical depth and significance of the research contribution, potential impact on theory and practice, and quality of presentation. A committee of individuals serving staggered three-year terms performs an initial screening to generate a short list, followed by an in-depth evaluation to determine the winning dissertation.

The selection committee will select the winning dissertation in early 2018.

Award

The Doctoral Dissertation Award is accompanied by a prize of \$20,000 and the Honorable Mention Award is accompanied by a prize of \$10,000. Financial sponsorship of the award is provided by Google.



Association for
Computing Machinery

DOI:10.1145/3131873

Developers know refactoring improves their software, but many find themselves unable to do so when they want to.

BY EWAN TEMPERO, TONY GORSCHKE, AND LEFTERIS ANGELIS

Barriers to Refactoring

REFACTORING⁶ IS SOMETHING software developers like to do. They refactor a lot. But do they refactor as much as they would like? Are there barriers that prevent them from doing so? Refactoring is an important tool for improving quality. Many development methodologies rely on refactoring, especially for agile methodologies but also in more plan-driven organizations. If barriers exist, they would undermine the effectiveness of many product-development organizations. We conducted a large-scale survey in 2009 of 3,785 practitioners' use of object-oriented concepts,⁷ including questions as to whether they would refactor to deal with certain design problems. We expected either that practitioners would tell us our choice of design principles was inappropriate for basing a refactoring decision or that refactoring is the right decision to take when designs were believed to have quality problems. However, we were told the decision of whether or not to refactor was due to non-design considerations.

It is now eight years since the survey, but little has changed in integrated development environment (IDE) support for refactoring, and what has changed has done little to address the barriers we identified.

We hope that presenting what we have learned will encourage improvement in refactoring support.

Reasons to Not Refactor

The reasons practitioners gave us for not refactoring a design believed to have quality problems can be categorized as follows:

Resources. Concern over the resources required was a frequently cited reason for not refactoring. The resource mentioned most often was time, as in "Deadlines often don't allow refactorings"; "Sometimes there is just no time"; and "No time no time."

Risk. Also frequently cited was the risk involved in making a change, in particular introducing new faults or other problems, as in "That kind of refactoring is time consuming and there is a large risk of introducing bugs" and "If it's working I leave it alone."

Difficulty. Another concern was the difficulty in making the change, as in "Inheritance is tricky to refactor correctly" and "This kind of refactoring is usually difficult."

ROI. Participants acknowledged that while there may be benefits from refactoring, there are also costs, and the return on investment, or ROI, has to be considered, as in "Again, I have to weigh the costs and benefits. The benefits have to be clear before taking on the costs of refactoring, retesting, etc."

Technical. Participants reported a variety of constraints due to characteristics of the project that restricted

» key insights

- **Developers understand the value of refactoring but are often prevented from doing it due to factors beyond their control.**
- **Refactoring has benefits, but also costs and risks, and developers' inability to quantify them inhibits their use of refactoring.**
- **The decision to refactor is ultimately a business decision, and support is needed to allow developers to make the business case regarding whether or not to refactor.**

ILLUSTRATION BY JUSTIN METZ

RESOURCES

TOOLS

RISK

ROI

SOURCES

TECHNOLOGY

DIFFICULTY

TOOLS

ROI

TECHNOLOGY

DIFFICULTY

RISK

their ability to refactor. Examples included having to implement a third-party interface that exceeds a limit, concern about the impact of any potential required changes on other parts of the system, the degree of familiarity with the code, dealing with legacy code, and lack of other support (such as test suites), as in “A large legacy codebase makes such refactoring tough”; “And with no tests, if I try to change things, they *will* break”; and “I will always refactor code I am creating; however, if that is not the case which is the most usual case then I don’t have time budgeted for refactoring.”

Management. Participants observed they did not always control what they spend their time on. Their managers or clients also had a say, as in “I want to, but the boss doesn’t like it”; “deadlines and pointy-haired bosses”; and “I’d like to more, but the client isn’t paying for it [sic].”

Tools. Inadequate tool support was mentioned as a reason for not refactoring; however, the tools referred to were not those that do the refactoring: “Source control makes this particular refactoring quite painful, that is, a lack of tool support.”

What Researchers Know

The results of our survey are still significant because they indicate missing support for refactoring. Although there has been much research on refactoring, including into developers’ use and factors that may affect their use, such studies have focused mainly on what developers do when they have decided to refactor, rather than whether or not they will refactor. For example, a 2006 study by Xing and Stroulia¹⁸ suggested refactoring was a common activity and raised questions about the quality of existing tool support. Murphy-Hill and Black¹¹ proposed five principles for supporting what they called “floss” refactoring: frequent refactoring intermingled with other kinds of program changes.¹¹ They also observed that existing tools rarely meet all principles and suggested this might explain the underutilization of tools they had observed. Consequently, tool-usability issues and their underuse is a recognized research focus.^{12,17}



Perhaps the most surprising theme is the complete lack of concern about the existence and quality of tool support for refactoring.



We know that developers refactor frequently but also that different developer groups have different refactoring behavior. Simple refactoring (such as the simple “rename” operation) is the most frequently applied. Generally, the more complex the refactoring operation is, the less frequently it is used.^{12,17}

Kim et al.¹⁰ studied the use of refactoring on Microsoft Windows, surveying more than 300 engineers who had carried out some form of refactoring of Windows, interviewing a subset. They identified such challenges performing refactoring as managing inter-component and inter-branch dependencies and maintaining backward compatibility.

More recently, Chen et al.³ surveyed 105 practitioners using agile software-development processes to understand how they planned and practiced refactoring, finding that while 98.1% of those surveyed did careful planning when creating new features, only 40% addressed refactoring.

Existing research has studied the behavior and issues faced by developers who have made the decision to refactor. Here, we are interested in those developers who have made the decision *not* to refactor.

Survey Design

We used a survey to increase the number of developers we could reach, looking to determine when and if refactoring is used. But instead of asking about this outright we used realistic scenarios as a basis for the investigation. The survey participants were doing development in some object-oriented language. We asked what they consider to be good design principles and later in the survey if these good principles were broken, would they use refactoring to fix the problem (or not). We asked them to indicate what they thought was a good design, focusing on a reasonable level of class depth and size.

The use of a survey limits the kind of questions we could ask, so we chose two simple design heuristics as our focus. We asked about the basis for two of the metrics from the well-known set of object oriented metrics proposed by Chidamber and Kemerer,⁵ namely Weighted Meth-

ods per Class, or WMC, and Depth of Inheritance Tree, or DIT. WMC is a form of size measure for classes, and we used its simplest form, a count of the number of methods in the class. DIT, which captures a characteristic of the inheritance hierarchy, is the length of the longest path from a class to a root of the hierarchy.

In the software engineering research community, these metrics are regularly presented as indicating possible design problems; the higher the values of WMC or DIT the more likely there are problems. Researchers have sought to establish thresholds for these and other metrics, including Chidamber et al.⁴ and Shatnawi.¹⁵ However, measurements of software systems indicate there are classes that have many methods or are very deep in the hierarchy.² The disconnect between theory and practice could be due to practitioners not being aware of the theory or knowing the theory but believing it to be wrong, perhaps because the theory really *is* wrong. We previously reported results for part of our survey indicating that 12.0% (452) of participants had a preference for a limit on the number of methods, whereas 25.2% (952) indicated a preference for a limit on the depth of a class; for details see Gorschek et al.⁷ These results suggest a significant proportion of developers believe the theory.

We asked survey participants whether they thought there should be a limit to the number of methods or depth of classes and, if so, what the threshold should be. It is important to note we let participants give their own views of what is good or bad, then, and only then, asked whether or not they would refactor designs in which the number of methods or depth exceeded the threshold they themselves found to be relevant. We first asked them to indicate what they believed was a good threshold for size—WMC—and depth—DIT. We then asked if they would use refactoring to fix classes in light of their chosen limits. We also gave them the opportunity to provide free-text commentary on their chosen response. This commentary was revealing as to what factors affected participants' decision to refactor (or not).

Survey Questions

We asked our survey participants about practices relating to commonly taught design principles for object-oriented programming; the full survey and raw data is available on our website <http://sefolklore.com>. We designed the questions relevant to this article as two pairs of questions for two design metrics, one pair (Q14, Q16) relating to class size as measured by number of methods and the second pair (Q18, Q20) relating to depth of a class in an inheritance hierarchy. We also invited participants to provide a free-text comment to Q16 and Q20 relating to their response. The labels for the possible responses to the questions were used to report the results.

Q14. What do you think is the largest number of methods a class should have? (Choose one of the alternatives, and add a number in the textbox that replaces “N” in the text of the alternative you chose.)

Never. There should never be more than about N methods.

Try. I try to avoid having more than about N methods in a class but allow exceptions in extreme circumstances.

Prefer. I prefer to avoid having more than N methods in a class but am not fanatical about it.

Don't. I don't really think about how many methods there are in a class but prefer to avoid having classes with more than N methods.

None. I don't think there should be any limit on the number of methods in a class.

Q16. What is the likelihood you will refactor a class (to create classes with fewer methods) if it has more methods than the limit you chose in question Q14?

Always. Always.

Most. Most of the time.

Obvious. Only if it is obvious how to refactor the class.

Nothing. Only if I have nothing else to do.

Forces. Only if someone forces me to.

Hardly. Hardly ever.

Q18. What do you think the maximum depth of a class should be in the inheritance hierarchy? (Choose one of the alternatives and add a number in the textbox that replaces “N” in the text of the alternative you chose.)

Similar responses, as with Q14, but replacing “number of methods” with “depth of class.”

Q20. What is the likelihood you will refactor the class hierarchy (to reduce the maximum depth of classes) if there are classes deeper than the limit you chose in question Q18?

Same responses as with Q16.

We analyzed the data using content analysis.¹³ Specifically, we coded the free-text answers using provisional coding¹⁴ to map statements to two main categories of general motivation for “why” or “why not” refactoring was done. Subsequently, we coded the overall categories using simultaneous coding¹⁴ to further refine types of motivation, as with, say, more detailed reasoning in terms of time, risk, or economic factors.

Survey Summary

We had 3,785 participants complete the compulsory questions of the survey (see the sidebar “Survey Questions”). The responses of greatest interest to us were from participants who indicated strong agreement with design principles that limit the number of methods or depth of a class. This is the category of developers who should be most interested in making sure that size and depth are not too large and deep, respectively, and so

would presumably consider refactoring. We used the responses “Never” or “Try” to Q14 (for methods) or Q18 (for depth) as indicating such strong agreement. We know (as reported in Gorschek et al.⁷) that 452 (12.0%) participants indicated a strong preference for establishing a limit on the number of methods, whereas 952 (25.2%) indicated a strong preference for there being a limit on the depth of a class. The median choice for the limit on number of methods was 10 and for depth 3.

Table 1 reports the distribution of responses from all participants to the questions on the likelihood of refactoring a class that exceeds a specified number of methods (Q16) or depth (Q20). And Table 2 reports the distribution of responses to Q16 and Q20 for just the “believers” who indicated strong agreement with the design principles. The columns “Comments for Q16” and “Comments for Q20” report, for those who would impose a limit, how many provided a free-text comment.

We classified participants responding “Always” or “Most” to questions Q16 and Q20 as an indication that they would refactor classes they perceived as being “bad.” Considerably more (952) agreed with having a limit on the depth of a class than on the number of methods (452). For those participants inclined to limit the number of methods, 265/452 (58.6%) would refactor classes that exceed their choice of limit (in bold). For those inclined to limit the depth of a class, 364/952 (38.2%) would refactor if the choice of depth was exceeded.

So developers are more inclined to limit the depth of a class than the number of methods but less inclined

to do anything when that limit is exceeded. In both cases a significant proportion would not refactor their designs even when, by their own assessment, a particular design had problems. We wanted to know why this was so.

One possibility is there is something about the participants’ backgrounds that influences their decision to refactor or not. For example, project managers may be more concerned about resources and risk, and inexperienced developers may view refactoring as too difficult. We collected demographic information for all participants, including amount of experience, programming language,

operating system, role, type of development, and highest qualification. This information indicated our sample provided a broad representation of developers. Half of the 3,785 participants reported more than four years of experience, half had a bachelor’s, and another 30% had master’s or higher academic credentials. Within the role category, almost all reported having a programmer role (95.6%), but all other roles also had good representation; for example, architects made up 64.1% of the total. Participants could report more than one role. The highest reported language was C# (55.7%), though Java (49.4%) and C++ (45%) were also well represented; full details are available in Gorschek et al.⁷

We examined just those participants who agreed with having a limit, as they were responding to whether they would refactor to deal with a design issue. Our statistical analysis, which was based on logistic regression, showed two groups—architects and experienced C# programmers—were more inclined to refactor if the number of methods exceeded the participants’ chosen limit. If the class depth exceeded the limit, C# developers and architects were again more inclined to refactor. Those in the programmer role were less inclined to refactor. Given that almost all participants indicated having a programmer role, this result may simply reflect the overall response.

These results are interesting, although none of the variables are good predictors, as they only weakly indicate whether or not developers will refactor. Neither participant role nor background gave a clear answer as to why they chose not to refactor even if their class design contradicted their view of good design. We had to dig deeper into the motivation offered by the respondents.

Participant Comments

To better understand what influences a decision to refactor, we examined the comments provided by those participants who indicated there should be a limit but would not refactor if the limit was exceeded. As outlined in Table 2, 162 participants who would limit the number

Table 1. Likelihood of refactoring a class if limit of number of methods or class depth is exceeded (all participants).

	Q16 Refactor Methods		Q20 Refactor Depth	
Always	125	3.3%	120	3.2%
Most	845	22.3%	536	14.2%
Obvious	1,725	45.6%	1,660	43.9%
Nothing	415	11.0%	366	9.7%
Forces	128	3.4%	224	5.9%
Hardly	547	14.5%	879	23.2%
Total	3,785		3,785	

Table 2. Likelihood of refactoring by those who limit number of methods or class depth.

Response to Q16/Q20	Q14 Limit Methods		Comments for Q16		Q18 Limit Depth		Comments for Q20	
Always	38	8.4%	14	8.6%	76	8.0%	29	8.4%
Most	227	50.2%	84	51.9%	288	30.2%	102	29.7%
Obvious	145	32.1%	50	30.9%	415	43.6%	138	40.1%
Nothing	28	6.2%	7	4.3%	77	8.1%	21	6.1%
Forces	4	0.9%	2	1.2%	26	2.7%	12	3.5%
Hardly	10	2.2%	5	3.1%	70	7.4%	42	12.2%
Total	452		162		952		344	

Table 3. Comment category for those who would not refactor (“No”) and those who would refactor (“Yes”); participants may be in more than one category.

Category	Limit Methods			Limit Depth		
	No		Yes	No		Yes
Resources	23	47.9%	27	49	36.0%	8
Risk	19	39.6%	6	43	31.6%	1
Technical	11	22.9%	11	25	18.4%	17
Difficulty	3	6.2%	5	13	9.6%	6
ROI	0	0%	0	13	9.6%	3
Management	1	2.1%	0	11	8.1%	3
Tools	1	2.1%	1	0	0%	0
Participants	48		46	136		35

of methods provided comments (“Comments for Q16”), as did 344 who would limit the depth of classes (“Comments for Q20”).

We identified the barriers (described earlier under the subhead “Reasons to Not Refactor”) through analysis of free-text comments. Not all comments explained participants’ reasons for not refactoring. The distribution of those who did is outlined in Table 3 (“No” columns). Some participants’ comments fall into more than one category; for example, we classified the participant comment “Unless a class is ridiculously huge or easy to refactor, it’s probably not worth the time or risk” as both “resources” and “risk.”

While we were most interested in participants who would not refactor, we note that those who *would* refactor made similar comments. The numbers in each category are also reported in Table 3 (“Yes” columns).

Interpreting the Results

As with any survey data, especially qualitative data based on free text, we urge caution in its interpretation. Our categories are not necessarily orthogonal; a management decision to not refactor might be due to perceived cost, risk, or ROI calculation. Some technical concerns may be about risk, as when the reason given for not refactoring is that there are insufficient tests, the real concern may be due to the risks. Some management restrictions could be due to managers’ concern over ROI, while concern regarding difficulty could be about the risk of introducing faults. Where we could, we chose the most specific category. Nevertheless, while the exact numbers may be uncertain, the themes are quite clear.

Perhaps the most surprising theme is the complete lack of concern about the existence and quality of tool support for refactoring. As other studies have shown,^{12,17} such concerns are indeed real. We speculate they come into play only when the decision to refactor has been made, though in our survey the quality of tools did not appear to be a factor in making that decision. This suggests the emphasis within the research community on the quality



The results indicate there is often little or no knowledge about what the actual ROI is when refactoring in a specific case.



of tool support is perhaps missing a significant issue.

Another theme we noted was that participants try to avoid the problem in the first place, as in “Would hope to never get to the case where a refactor of this type is necessary” and “This should be caught at design time. I am cautious to refactor this problem.” One participant made this comment regarding “number of methods,” and 16 participants made it for “depth of class.” So there is awareness of the benefits of good design. However, even with the best of intentions, design quality can degrade, so avoidance is no solution and poor designs still need refactoring.

While our results suggest the removal of perceived design problems is not a priority, many responses indicated awareness of the benefit of refactoring, as in “Time taken to refactor the code is probably a good investment, since it will probably save developer time later.” However, a common theme was that external factors (such as “deadlines” and “perceived risk”) led to the decision to *not* refactor, as in “Unless a way to refactor is clear, time pressure generally prevents refactoring” and “Refactoring class hierarchies is usually really time-consuming and can have all kinds of unintended effects.” Implicit in such comments is the suggestion that the benefits of refactoring do not always outweigh the cost—that refactoring is not always a good investment, as in “It sometimes takes more effort than it is worth to reduce the depth.”

Whether such investment is worth it was more explicitly mentioned in other responses, as in “Again this is the time trade-off of do I see a clear opportunity to improve things and how long will it take?” In particular, the inability to determine ROI is a barrier, as in “If I can’t make a reasonable estimation on how long it will take, I will leave it alone.” That is, the decision to not refactor was not due to it being considered a bad idea but to uncertainty as to the benefit. The results indicate there is often little or no knowledge about what the actual ROI is when refactoring in a specific case.

As mentioned earlier, studies indicate developers prefer simple refac-

toring. For example, Murphy-Hill et al.¹² asked “Why is the ‘rename’ refactoring tool so much more popular than other refactoring tools?” The answer could be simply that the “rename” operation and less-complex operations are, in general, what developers need most often. However, our results suggest this is not the case. We see that developers do want to refactor complex cases and suggest that the preference for a simpler operation can be explained in ROI terms; that is, being simple, the cost is low, meaning it is almost always worth doing. More-complex operations take longer, are higher risk, and deliver benefit that is not so clear, as when one respondent said, “I will refactor the class only if it has a clear benefit.”

What We Learned

We asked whether developers would refactor classes that did not meet characteristics they themselves thought were important, finding many (at least 40%, as in Table 2) would not. The reasons can be summarized as lack of resources, of information identifying consequences, of certainty regarding risk, and of support from management. This is consistent with studies on tool use. In particular, Vakilian et al.¹⁷ reported that reasons for not using automation for refactoring involve trust, predictability, and complexity.

What we did not see was concern over lack of tool support. This was surprising, since, as Murphy-Hill and Black¹¹ noted, two advantages from using refactoring tools are lower error rates and less time required, so good tool support should go some way toward addressing developers’ concerns.

Recent tool research has focused on primitive refactoring operations—“rename,” “move,” “method,” and so on—whereas our question was goal-directed, asking, “Will you refactor to achieve this effect?” rather than simply, “Would you use this refactoring operation?” We speculate that an unstated barrier is the difficulty translating a refactoring goal into refactoring operations. While research has proposed refactoring operations to remove code smells (such as described by Bavota et al.¹), Vakilian et



We speculate that an unstated barrier is the difficulty translating a refactoring goal into refactoring operations.



al.’s survey¹⁷ identified a preference for lightweight methods for invoking refactorings. We suggest providing lightweight goal-directed refactoring recommendations would be a fruitful area for future research.

We also did not see lack of interest in refactoring. Our participants were by and large willing to consider refactoring, but barriers prevented them doing so. Tool support for performing refactoring can address some of them but not all. For example, lack of certainty about ROI is a challenging but also possibly very rewarding area of research. We are not the first to highlight the importance of using ROI (see, for example, Harun and Lichter,⁸ Kazman et al.,⁹ and Szöke et al.¹⁶), but our results suggest it is a notable barrier to improving design quality. What is needed is a decision-support system that allows practitioners to be able to quantify benefit in the long and short term. This would help inform the decision as to whether the required resources are justified or the potential risk is tolerated. It would also allow developers and managers to make informed choices as to whether or not to refactor.

The reasons participants gave for not refactoring will mostly come as no surprise, especially to practitioners. What we provide is solid data to back up everyone’s suspicions. Practitioners who have been prevented from refactoring for similar reasons can at least take heart that they are not alone, or even, it may seem, in the minority. While existing research goals regarding refactoring are reasonable, we feel they do not address problems actually faced by practitioners. As researchers, we need to better understand what barriers they face and better target our research to support them.

Validity of Results

Our data is representative because of the way we elicited it. We avoided leading questions, especially for free-text responses, asking only, “Please explain your answer.” This led to comments that did not provide specific reasons for refactoring (or not). A more targeted question might have yielded more and clearer results but might also have biased the responses.

None of the material we used to recruit participants mentioned refactoring or gave specifics of the concepts we would be asking about. Consequently, unlike other studies, rather than include only developers known to refactor, there should have been no obvious bias for or against the use of refactoring. We recruited through personal contacts, word of mouth, and social media, supported by our website <http://se-folklore.com> and a YouTube video. The demographic data we collected reflected a variety of experience, roles, languages, level of qualification, and type of development.

We set a high bar in assessing whether a participant would agree with placing a limit on number of methods or class depth or would still refactor when that limit is exceeded. Different choices would change the distribution of the frequencies in the different categories somewhat, but the same trends would be evident.

Our wording of the possible responses might have affected what responses participants chose, but the barriers we identified came from the free-text commentary, which was unlikely to be affected by such concerns. Even those who would refactor mentioned some of the same concerns (as reported in Table 3 “Yes” columns). Participants from all demographics provided comments, so there is no obvious bias due to background, as was also the case with our target group—those who would limit number of methods or depth of inheritance.

We conducted the survey more than eight years ago and IDEs have since improved, so perhaps we would get different results today. However, little has changed in refactoring support, and we are not aware of any research directly addressing the issues our survey identified. More recent surveys have identified similar issues. For example, Kim et al.¹⁰ reported concerns regarding lack of adequate test suites. Chen et al.³ noted 21% of participants reported lack of support from management as a reason for not refactoring and 12.4% did no planning regarding refactoring. However, these studies surveyed developers who were already committed to refac-

tor, whereas ours included those who chose not to refactor. That these more recent surveys report findings similar to ours suggests there has been little change in barriers to refactoring since our survey.

We used two metrics to identify potential design-quality issues based on published theories. It is possible the theories are wrong. However, our conclusions are based on comments by participants who (rightly or wrongly) believed the theories.

Conclusion

There are significant barriers preventing developers from refactoring to remove software design-quality issues, no matter how they are identified. Reducing or even eliminating the barriers has the potential to significantly improve software quality. One means is to provide refactoring support that is goal-directed rather than operations-directed. Another is to provide better quantification of the benefits, thus better informing the decision as to whether or not to refactor.

Acknowledgments

We thank our survey participants, many of whom made the extra effort to provide the comments we included and discussed here. We also thank the anonymous reviewers for their valuable comments and suggestions. 

References

1. Bavota, G., Oliveto, R., Gethers, M., Poshyanyk, D., and De Lucia, A. Methodbook: Recommending move method refactorings via relational topic models. *IEEE Transactions on Software Engineering* 40, 7 (July 2014), 671–694.
2. Baxter, G., Frean, M., Noble, J., Rickerby, M., Smith, H., Visser, M., Melton, H., and Tempero, E. Understanding the shape of Java software. In *Proceedings of the 21st Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications* (Oct. 2006), 397–412.
3. Chen, J., Xiao, J., Wang, Q., Osterweil, L.J., and Li, M. Perspectives on refactoring planning and practice: An empirical study. *Empirical Software Engineering* 21, 3 (June 2016), 1397–1436.
4. Chidamber, S.R., Darcy, D.P., and Kemerer, C.F. Managerial use of metrics for object-oriented software: An exploratory analysis. *IEEE Transactions on Software Engineering* 24, 8 (Aug. 1998), 629–639.
5. Chidamber, S.R. and Kemerer, C.F. A metrics suite for object-oriented design. *IEEE Transactions on Software Engineering* 20, 6 (June 1994), 476–493.
6. Fowler, M. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, Boston, MA, 1999.
7. Gorschek, T., Tempero, E., and Angelis, L. A large-scale empirical study of practitioners' use of object-oriented concepts. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering* (Cape Town, South Africa, 2010), 115–124.

8. Harun, M.F. and Lichter, H. Towards a technical debt-management framework based on cost-benefit analysis. In *Proceedings of the 10th International Conference on Software Engineering Advances* (Barcelona, Spain). International Academy, Research, and Industry Association, 2015.
9. Kazman, R., Cai, Y., Mo, R., Xiao, L., Feng, Q., Haziye, S., Fedak, V., and Shapochka, A. A case study in locating the architectural roots of technical debt. In *Proceedings of the 37th International Conference on Software Engineering* (Firenze, Italy). IEEE Press, 2015.
10. Kim, M., Zimmermann, T., and Nagappan, N. A field study of refactoring challenges and benefits. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering* (Cary, NC). ACM Press, New York, 2012, article 50.
11. Murphy-Hill, E. and Black, A.P. Refactoring tools: Fitness for purpose. *IEEE Software* 25, 5 (Sept.-Oct. 2008), 38–44.
12. Murphy-Hill, E., Parnin, C., and Black, A.P. How we refactor, and how we know it. *IEEE Transactions on Software Engineering* 38, 1 (Jan. 2012), 5–18.
13. Robson, C. and McCartan, K. *Real World Research, Fourth Edition*. John Wiley & Sons, Inc., 2015.
14. Saldana, J. *The Coding Manual for Qualitative Researchers, Third Edition*. SAGE Publications Ltd., 2016.
15. Shatnawi, R., Li, W., Swain, J., and Newman, T. Finding software metrics threshold values using ROC curves. *Journal of Software Maintenance and Evolution: Research and Practice* 22, 1 (Jan. 2010), 1–16.
16. Szöke, G., Nagy, C., Ferenc, R., and Gyimóthy, T. A case study of refactoring large-scale industrial systems to efficiently improve source code. In *Proceedings of the 14th International Conference on Computational Science and Its Applications* (Guimarães, Portugal, June 30–July 3). Springer International Publishing, Cham, Switzerland, 2014, 524–540.
17. Vakilian, M., Chen, N., Negara, S., Rajkumar, B.A., Bailey, B.P., and Johnson, R.E. Use, disuse, and misuse of automated refactorings. In *Proceedings of the 34th International Conference on Software Engineering* (Zurich, Switzerland). IEEE Press, 2012, 233–243.
18. Xing, Z. and Stroulia, E. Refactoring practice: How it is and how it should be supported (an Eclipse case study). In *Proceedings of the 22nd International Conference on Software Maintenance* (Philadelphia, PA). IEEE Press, 2006, 458–468.

Ewan Tempero (e.tempero@auckland.ac.nz) is an associate professor of computer science at the University of Auckland, Auckland, New Zealand.

Tony Gorschek (tony.gorschek@gmail.com) is a professor of software engineering at the Blekinge Institute of Technology, Karlskrona, Sweden.

Lefteris Angelis (lef@csd.auth.gr) is a professor of statistics and information systems in the Department of Informatics at Aristotle University of Thessaloniki, Thessaloniki, Greece.

©2017 ACM 0001-0782/17/10



Watch the authors discuss their work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/barriers-to-refactoring>

DOI:10.1145/3132745

Millennials entering the workforce ignore the risks of using privately owned devices on the job.

BY HEIKO GEWALD, XUEQUN WANG, ANDY WEEGER,
MAHESH S. RAISINGHANI, GERALD GRANT,
OTAVIO SANCHEZ, AND SIDDHI PITTAYACHAWAN

Millennials' Attitudes Toward IT Consumerization in the Workplace

PEOPLE BORN AFTER 1980, often called “millennials” by demographic researchers, behave differently from older generations in significant ways. They are the first “digital natives,” the “always on generation” that expects to have information instantly and always available at its fingertips. Their attitudes have been described by previous research in often unfavorable terms. And when they enter the workplace, they pose a major challenge to managers from older generations, who, it has been shown, typically follow a different set of values.

Our research investigates the attitudes of millennials who have not yet entered the workforce toward the use of information technology (IT) in terms of “IT consumerization.” Specifically, we want to know how this significant part of the population

weighs benefits against risks when it comes to intention to use technology in a business environment.

In 2013, we conducted an international study involving 402 students in their final year of undergraduate study just before entering the workplace. We received feedback from students at Neu-Ulm University of Applied Sciences (Germany), Dongbei University of Finance & Economics (China), Texas Woman's University (U.S.), Carleton University (Canada), Fundação Getúlio Vargas (Brazil), and RMIT University (Australia). We found they share a common set of values regardless of nationality, including motivational drivers that would alarm corporate IT managers, if known. The individuals in our sample value their own benefit highly and dramatically neglect the risks their actions might pose.

The way we work, think, and behave is heavily influenced by the Internet, email, smartphones, and other technological innovations that have proliferated over the past 20 to 30 years. The generation of people born after 1980 is the first to grow up with information everywhere, anytime²⁴ and referred to as digital natives, or, more commonly, millennials.¹⁵

Many studies have sought to analyze them.²⁰ For example, in their 2010 literature review, Ng et al.²⁴ characterized them as “want it all” and “want it now.” It seems generally accepted in research and practice alike that millennials are difficult to cope with

» key insights

- **Members of Generation Y (so-called millennials) see the use of their privately owned devices for work as a necessity, not an option.**
- **Millennials focus on their personal benefit, generally ignoring the risks they may introduce into corporate networks when using their own devices and accounting for risk only if it threatens them directly.**
- **When it comes to weighing risks against benefits, such behavior is seen across developed economies, with no significant cultural influence across an international sample.**



when entering the workplace from the perspective of managers born perhaps decades earlier.³² However, given the need for talent in today's technology-driven society, especially in computer science, management needs to adapt and offer millennials working conditions that attract them.³⁷

Previous research focused mainly on millennials' attitudes toward work.³ Their attitude toward using technology for work purposes has not yet, however, received sufficient academic attention.³⁵ There is a striking paucity of research looking at how millennials use technology for professional and personal purposes.³⁵ To partially close this gap, we conducted our study on the motivational factors that shape millennials' intention to use technology in the context

of IT consumerization. Here come the cherry pickers.

Research Background

Here, we introduce the concept of IT consumerization, then discuss how national culture was relevant in our study:

IT consumerization. IT consumerization describes the ongoing process of blending private and business life when it comes to the use of technology, as driven by employees who push IT solutions they use privately into the workplace.³⁸ This applies to hardware (as in “bring your own device,” or BYOD) where privately owned laptops, tablets, and smartphones are used for business tasks but also to software when online email services or cloud-storage solutions are used for busi-

ness purposes.³¹ “Ownership” of the device or service is usually regarded as a key characteristic of IT consumerization,^{14,25} as possession shifts from employer to user/employee.

We expect IT consumerization to have a positive influence on employees' work performance by increasing satisfaction, flexibility, and mobility.¹⁴ It is also demanded by more and more employees who want to use their own smartphone to, say, access corporate email messages.³⁶ However, using privately owned devices at work involves risks like blurred boundaries between professional and private lives,^{5,13,25} creating additional stress for the employee (such as when responding to email messages on weekends when technically not at work). Not only users, but their employers as well, face notable

challenges from this trend. Anecdotal evidence indicates corporate IT departments are under pressure to give in to user demands to be allowed to use privately owned IT for work purposes. However, granting myriad different consumer devices access to the corporate network is a nightmare for anyone concerned about IT security.

Consumerization of IT seems to be a key characteristic of millennials, as their desire to be always on is not limited only to the workplace. In the same vein, they are accustomed to always using state-of-the-art technology, something not every work environment is able to provide, especially because the definition of “state of the art” is subjective.

Our study focused on mobile devices as an exemplary technology to explain millennials’ behavioral intentions when it comes to the use of technology. This area is of great concern to practitioners, including CIOs and senior IT managers.³¹

To understand the role of such contradictory factors in individual decision making, social psychology provides net-valence models (NVMs) that assume individuals intend to perform an action only if the perceived benefits outweigh the associated costs.^{7,26} Prior research found NVMs help explain the adoption of technology-related services.¹⁷ Other prominent theories on technology adoption (such as Unified Theory of Acceptance and Use of Technology³³) do not capture the risks associated with technology use and is why we chose NVMs as our theoretical lens (see Figure 1).

Cultural values and IT use behavior. Millennials’ use of corporate IT involves multiple challenges for IT executives worldwide. However, the literature suggests behavioral models do not apply universally across all cultures.³⁰ Research by Srite and Karahanna³⁰ showed the significance of factors determining technology use are notably dependent on espoused cultural values, particularly those reflecting national culture. National culture refers to “the collective programming of the mind that distinguishes the members of one group or category of people from another.”¹¹ Hofstede and Bond¹⁰ proposed five dimensions of national culture: power distance;

individualism/collectivism; masculinity/femininity; uncertainty avoidance; and long-term orientation. Other research found that national cultural values strongly affect an individual’s IT-adoption behavior.^{12,32}

In order to understand how millennials perceive benefits and risks associated with IT consumerization across different cultures, we identified uncertainty avoidance (UA) and individualism/collectivism (IC) as the most relevant dimensions. Power distance, masculinity/femininity, and long-term orientation are important in the more general context of technology adoption but less relevant in understanding the effect of perceived risks and benefits in the context of our study.

“Uncertainty avoidance” refers to “the extent to which individuals feel vulnerable to unpredictable and unknown situations.”⁹ People with strong UA values fear uncertainty. In the context of work-related technology, they need the predictability often provided by rules, policies, and structure in organizations that IT consumerization contradicts or dilutes. UA can thus help understand how millennials perceive the risks associated with IT consumerization.

“Individualism/collectivism” is one of the most widely studied cultural values in cross-cultural research,³⁰ referring to “an individual’s preference for a social framework where individuals take care of themselves (individualism), as opposed to how they expect the group to take care of them in exchange for their loyalty (collectivism).”⁹ Individuals with individualistic values have a more complex and more frequently sampled private self. Consequently, their own goals, beliefs, and values are more salient. Considering technology use at work, they are more concerned with the benefits they might achieve than the disadvantages that could arise for others. IC is thus useful in understanding how millennials perceive benefits associated with IT consumerization, especially when there could be conflict between themselves and their employers.

Research Model

Based on NVMs, it is assumed an individual’s behavioral intention to use

privately owned devices on the job is determined by the outcome of weighing perceived benefits vs. perceived risks/associated costs, as in Figure 1.

Perceived benefits. Perceived benefits “include all benefits which the customer perceives as having been received.”¹⁸ In the context of IT use, they reflect the overall positive utility individuals expect when using a particular technology.¹² Prior research demonstrated that perceived benefits significantly affect behavioral intention regarding the use of IT.^{16,19}

We define perceived benefits as individuals’ assessment of the functional benefits they associate with using a privately owned device for work purposes. Building on the premises of technology acceptance and use models,^{22,29,33} we propose that the benefits of using a privately owned device for work purposes are related to the characteristics of the technology and the functional advances it provides.²⁹ We thus assume perceived benefits as a multidimensional construct comprising three facets of employment behavior: performance expectancy; effort expectancy; and compatibility.

Employees may realize productivity gains when allowed to select devices on their own.¹⁴ Consequently, performance expectancy reflects the extent individuals perceive that using privately owned devices supports their ability to perform better at work.³³ Moreover, devices selected by individual employees are usually perceived as easier to use and more intuitive than those provided by an IT department.²⁵ We thus define effort expectancy as the degree of ease an individual associates with using a privately owned device as compared to using a device provided by an IT department. Overall benefit perceptions are also formed by an individual’s work style and associated needs and values. To capture these influential factors, Moore and Benbasat²² proposed the construct “compatibility” as the degree to which using a privately owned device for work purposes fits the individual’s work style. Employees who agree to be available for work responsibilities (such as to respond to email messages) after work hours are more likely to see the use of their devices for business purposes as beneficial.

Following these insights, we hypothesize that perceived benefits influence individuals' consumerization behavior:

Hypothesis 1. The greater the perceived benefits of using privately owned devices for work purposes, the greater an individual's intention to participate in a BYOD program.

"Perceived risk" reflects negative utility from a subjective perspective, a concept introduced by Bauer² as part of his "Perceived Risk Theory," which assumes subjective risk perceptions directly influence an individual's intention to perform a certain action.⁴ Perceived risk is defined by Cunningham⁴ as "the amount that would be lost, or that which is at stake, if the consequences of an act were not favorable, and the individual's subjective feeling of certainty that the consequences will be unfavorable." Featherman and Pavlou⁶ and Hoehle et al.⁹ found perceived risk plays a significant role in individuals' IT-use behavior.

To reflect the perceived cost associated with using privately owned devices, we define perceived risk as the belief of individuals about the potential negative outcomes caused by using privately owned devices on the job. The negative consequences of such behaviors can be classified into multiple types of loss, indicating that, as with perceived benefit, perceived risk is a multidimensional construct.^{6,16} Based on the arguments discussed earlier regarding consumerization and its effects on corporate IT, we hypothesize that using privately owned devices for business purposes encompasses three facets of risk: performance; privacy; and security.

Using privately owned devices for work purposes generally shifts responsibility from the IT department to the individual. For instance, the individual is, at least psychologically, accountable for "how well the [device] will perform relative to expectations."¹ The risk associated with using one's own devices on the job includes the potential that the device the individual is responsible for is not sufficient for its intended business purpose. Performance risk thus reflects the potential for not being able to perform business activities as expected.



Granting myriad different consumer devices access to the corporate network is a nightmare for anyone concerned about IT security.



Using a device for both private and business purposes entails the risk that personal information is disclosed to the employer without the employee's consent and knowledge.²¹ Privacy risk, as defined by Featherman and Pavlou⁶ as the "potential loss of control over personal information,"⁶ encompasses this facet of risky behavior. Business data, as well as personal data, is at risk. The potential for corporate data to be exposed to unauthorized third parties also increases when individuals use their private devices for work purposes.²⁵ Information security is one of the most important topics related to IT consumerization, as 90% of all corporate data breaches fall into four patterns:³⁴ lost and stolen devices, user-initiated crimeware, insider misuse, and miscellaneous human errors. To capture this facet of risky behavior, we assume security risk, or potential loss due to fraud or a hacker compromising corporate information security,¹⁶ contributes to overall perceived risk.

We thus hypothesize that perceived risk negatively affects individuals' decisions regarding use of privately owned devices at work:

Hypothesis 2. The greater the perceived risk of using privately owned devices for work purposes, the lower an individual's intention to participate in a BYOD program.

We also assume the perceived risk associated with IT consumerization influences behavioral intention indirectly by negatively affecting perceived benefits. For instance, as a measure of safeguarding IT security, firms usually adopt policies that allow them to erase data when an employee's device is lost or stolen. Such "loss of full ownership" significantly affects the perceived benefits of BYOD.²⁸ We thus propose:

Hypothesis 3. The perceived risk of using privately owned devices for work purposes negatively affects an individual's perception of benefit.

Cultural values. Research provides evidence that millennials' cultural values influence their technology-use behavior.³⁰ However, it remains to be demonstrated whether the proposed NVM holds across the general population of millennials who reflect a variety of cultural values.³⁰ We propose

Table 1. Dataset demographics.

	Cluster A	Cluster B	Complete Set C
	IC + / UA –	IC – / UA +	
n	242	160	402
Gender			
Male	44.2%	37.5%	41.5%
Female	55.4%	62.5%	58.2%
Age			
<= 21 years	53.7%	55.6%	54.5%
22 to 25 years	46.3%	44.4%	45.5%
Country			
Australia	9.9%	6.3%	8.5%
Brazil	7.0%	9.4%	8.0%
Canada	6.2%	10.6%	8.0%
China	39.7%	27.5%	34.8%
Germany	13.6%	13.8%	13.7%
U.S.	14.0%	23.8%	17.9%
Other	9.6%	8.6%	9.1%

of using privately owned devices for work purposes and hence to be more likely to use their own devices at work. We also expect perceived risk to be less important for millennials who espouse lower uncertainty-avoidance cultural values.

We thus propose that there are distinctive subcultures within the overall group of millennials that are determined by differences regarding their UA and IC values. We hypothesize that the subcultures will take different approaches toward risk/benefit-assessment of their intention to participate in a BYOD program:

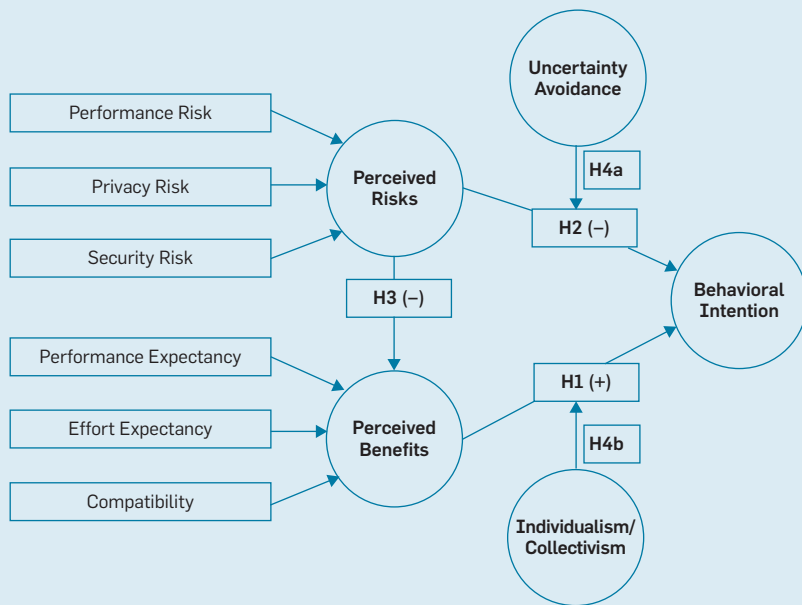
Hypothesis 4. The effect of the perceived risks of using privately owned devices for work purposes differs among millennials with lower UA scores and millennials with higher UA scores. The effect of perceived benefits of using privately owned devices for work purposes also differs among millennials with higher IC scores and millennials with lower IC scores.

Methodology

Here, we discuss our data collection, sample clustering, data analysis, and results:

Data collection. To test our research model, we developed a questionnaire with a set of measurement items for each construct, and to safeguard measurement validity, we adapted items from prior research, as outlined in the online appendix (dl.acm.org/citation.cfm?doid=3132745&picked=formats).

We distributed the questionnaire among students in their final year of undergraduate studies and with relevant work experience (most respondents worked full time for at least six months during their studies) using an online survey tool. Our approach is consistent with Vodanovich et al.³⁵ who suggested conducting surveys with students to understand how millennials (“digital natives” in their terminology) use technology. We collected data from students with “technology-affine” majors—“information systems,” “industrial engineering,” and “business administration”—in a number of universities worldwide. We chose countries with different values of UA and IC, according to Hofstede.¹¹ After stripping out incomplete ques-

Figure 1. Research model.

that the explanatory power of the theoretical model depends on how distinctively millennials espouse characteristic cultural values.

Based on these arguments, we expect to see distinctions in individualistic values and perceptions toward uncertainty. Using their own devices for work purposes enables millennials to express their sense of self and better achieve their own goals and follow their own beliefs and values. Also, us-

ing a privately owned device can indicate that individuals are more strongly concerned with their own needs than with those of the collective, including their employers. We thus expect individuals who espouse individualistic cultural values to more strongly value the benefits of using privately owned devices. Likewise, we assume individuals who experience less difficulty dealing with uncertainty to put less emphasis on the potential risks

tionnaires, we received a total of 402 valid responses.

Clustering for espoused cultural values. We conducted exploratory factor analysis, a statistical method used to uncover the underlying structure in a large set of variables, to test the “unidimensionality” of the measurement items for IC and UA. It revealed three items measuring IC and two items measuring UA load with a high coefficient on the factors they are intended to measure (loadings > 0.79). Using the factor scores of these items, we then conducted a K-means clustering. Cluster analysis revealed two clusters (see Table 1) where the first cluster (referred to as A) encompassed respondents with high IC scores (cluster center 0.13) and low UA scores (cluster center -0.61), and the second cluster (referred to as B) encompassed respondents with rather low IC scores (cluster center -0.20) and rather high UA scores (cluster center 0.93).

We characterized the respondents in group A as more individualistic and less risk-averse than in group B. Although both groups encompass only millennials, they showed different characteristics when it comes to the formation of behavioral intention. One could argue A is the more forthcoming, self-centered, and aggressive, while B represents the more group-oriented and considerate.

Measurement model assessment. We tested our model with partial least squares using SmartPLS 3.0 with 1,000 samples bootstrapping, assessing the measurement model with the complete dataset C, as well as with clusters A and B.

We measured behavioral intention reflectively (loadings of the indicators were above 0.95 and significant at the .001 level) and confirmed internal consistency by assessing Cronbach's Alpha (CA) and Composite Reliability (CR) measures. Both exceeded the threshold of 0.90 for all datasets: A CA=0.90, CR=0.95; B CA=0.92, CR=0.96; C CA=0.91; and CR=0.96. The average variance extracted was greater than 0.50 (A 0.91; B 0.93; and C 0.92), demonstrating sufficient convergent validity. Finally, we used cross-loading analysis to confirm that all constructs load highest with their respective items.

The formative measures of “perceived benefits” were significant, at least at the .05 level, and path coefficients were greater than .1, suggesting the chosen characteristics of each category were relevant for the formation of the construct (see Table 2). Moreover, the variance inflation factor (VIF) was less than 2, supporting our assumption for indicator validity.

The formative measures of “perceived risks” revealed mixed results regarding the risk facets' contribution to the formation of the formative index. We found privacy risk was not relevant regardless of dataset used; performance risk was significant only in subset B and the complete dataset;

and security risk contributed significantly only to the formative index of the complete dataset. Although not all facets of risky behavior contributed significantly, VIF was less than 2 within all three datasets, confirming indicator validity. Consequently, low redundancy of indicators' information was confirmed.

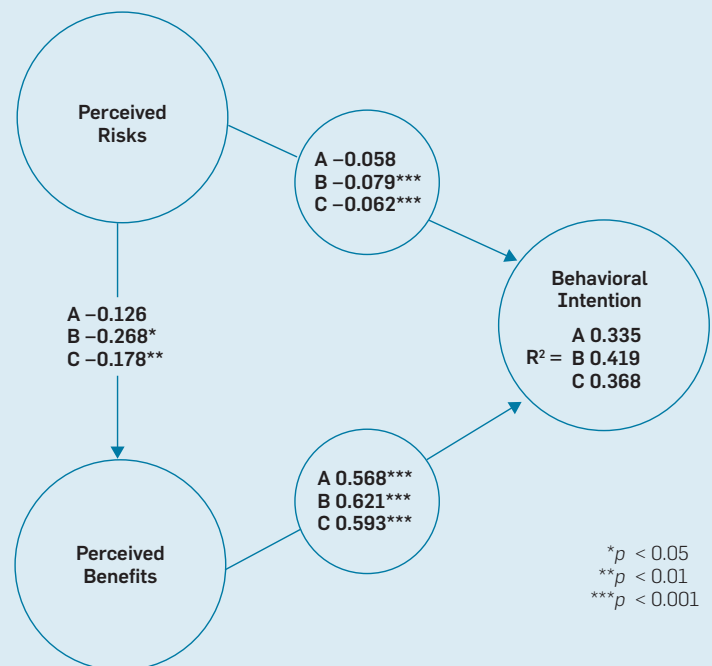
The results show performance risk contributed significantly to overall risk perception in dataset B and C, while security risk contributed to overall risk perception in only the complete dataset C. Privacy risk did not significantly contribute to perceived risk, regardless of dataset. And performance expectancy, effort expect-

Table 2. Formative constructs measurements.

Construct	Facet	Cluster A	Cluster B	Complete Set C
Perceived Risks	Performance	0.740	0.631*	0.713**
	Privacy	-0.315	0.133	-0.124
	Security	0.668	0.530	0.613*
Perceived Benefits	Performance Expectancy	0.522***	0.430**	0.491***
	Effort Expectancy	0.325*	0.230*	0.292***
	Compatibility	0.349**	0.538***	0.420***

*** $p < 0.001$; ** $p < 0.01$; * $p < 0.05$

Figure 2. Structural model assessment.



tancy, and compatibility contributed significantly to perceived benefits in all datasets.

Structural model assessment. The results (see Figure 2) demonstrate the millennials responding to the survey primarily consider the benefits and neglect the risks of using technology; in our research context, which involves taking part in a corporate BYOD program.

Detailed analysis confirmed the existence of two distinct groups of millennials differentiated by their espoused cultural values in our sample. We named them due to their strong characteristics the “narcissists” (group A) and the “prudent” (group B). Both groups showed intentions expected by millennials: take the benefits and ignore the risks. However, the groups also showed different approaches to acting on their risk-and-benefit perceptions, depending on their respective IC/UA scores.

For millennials with high IC scores and low UA scores—narcissists—risk perceptions have no effect, while benefits are weighted comparatively high when it comes to technology-adoption decisions. Additionally, perceived risks do not significantly affect perceived benefits. For the group with low IC scores and high UA scores (prudent) risk and benefit perceptions are—relative to the narcissists—more important for the intention to participate in a BYOD program. For these millennials, we found perceived risk was significantly related to perceived benefits (medium effect, Cohen’s $d=.163$), indicating they are more aware of risk and react more cautiously.

Overall, the results show our research model is capable of explaining a good portion of the variance in millennials’ behavioral intention to participate in a BYOD program ($R^2=.37$). The explanatory power of the model is slightly stronger ($R^2=.42$) for individuals categorized as the prudent.

Finally, removing perceived benefits from the model leads to a significantly stronger path between perceived risk and behavioral intention for the prudent ($\beta=-.244^{**}$) and for the combined dataset ($\beta=-.166^{**}$). This result indicates the effect of perceived risk on behavioral intention is most likely mediated by perceived benefits.

Discussion

These results confirm our expectations but also point to unanticipated conclusions. Millennials are egocentric, especially when it comes to their expectations regarding the workplace.^{24,27} The respondents in our survey expect great benefit from using their own devices, but the related risks are not reflected in their behavioral decision patterns unless they are personally affected. In our sample, we also found that only those millennials who are more strongly oriented toward the collective and somehow feel vulnerable in unpredictable situations (prudent) are influenced to at least some degree by risk perceptions regarding the performance of their privately owned devices. These findings confirm what research expects^{23,24,27} and anecdotal evidence underlines.

What we did not expect was the cross-cultural homogeneity of the sample. Millennials’ values seem uniform regardless of cultural background. Throughout our sample we did not find statistically significant differences in responses based on interviewee nationality. Investigating further, we used two distinct cultural values—IC and UA—to cluster the responses. We were able to distinguish between two groups—narcissist and prudent—that react as typical millennials but also differ in tendencies toward their intention to participate in BYOD programs. Forming their intention, the prudent (with lower IC and higher UA) seem slightly more risk-sensitive than the narcissists (with higher IC and lower UA). Contrary to our expectations, the two groups were well distributed among all participants in our sample; that is, we found no significant correlation with nationality or any other control factor to define the two clusters.

Even if the intention to use privately owned devices had formed differently depending on IC and UA scores, millennials remain millennials first and foremost. The proposed facets of risk do not significantly contribute to their risk perceptions, which, in turn, do not show considerable direct effect on behavior intention to participate in corporate BYOD programs. Our data implies that risk perceptions influence their decision making, which is

mediated through perceived benefits. However, this effect applies only to group B.

That is, the results show that the espoused cultural values individualism and uncertainty avoidance only slightly influenced survey-responding millennials’ decision making regarding their IT consumerization behavior. Given these findings, our study contributes to theory and practice alike. First, it provides an NVM that accounts for the particularities of IT consumerization, demonstrating the risks/costs associated with the use of privately owned devices at work do not significantly affect decision making for most millennials. Further evidence is thus given that millennials are concerned mainly with their own benefit and happy to neglect the risks that do not jeopardize them directly. Privacy risk seems to be of no concern, as in Vodanovich³⁵

Second, the study also enhances our knowledge of the role of millennials’ espoused cultural values, showing for the first time in the academic literature that the formation of technology-use decisions are apparently universal for millennials independent of cultural background.

For corporate IT managers, our findings are a warning: Millennials will quite likely use their private devices for work, not necessarily as part of an official program, then through shadow IT. To avoid this, structured offerings should be made available. IT security managers should thus consider the implications of millennials’ egocentric approach to risk. If users are not concerned about the risks their behavior poses to the company, then the company itself must be doubly cautious. To counter this potential threat, companies need to create robust BYOD programs that provide employees the benefits of using their own devices and simultaneously safeguard corporate data and networks.

Our findings embrace the typical limitations of empirical research: We had only a limited number of participants from a limited number of countries and who cannot be representative of the global millennial population. As such, our findings represent an indication but are not generalizable for all millennials. Additionally, our

research subjects were students. Although all participants belonged to the millennial generation and had working experience, they were currently not all actually working. The part of the millennials population already in the active workforce was not included in our dataset. Even though we took care, it cannot be ruled out that the occasional participant lacked the required work experience. It is thus possible that some participants did not fully understand or misinterpreted the BYOD concept and its implication for their future working lives.

Conclusion

We conducted an international study to obtain information about how millennials about to enter the workforce weigh risks and benefits when it comes to using privately owned technology in the workplace. It included 402 students in their final year of study and with relevant work experience from six countries. The results indicate they pay a lot of attention to their own benefit and significantly neglect the risks associated with using privately owned technology on the job. We tested our hypothesis by asking them about their intention to use privately owned devices at work by enrolling in a corporate BYOD program. The responses showed they expect to use their private devices for work purposes. They perceive major benefits from using their own devices but pay little attention to the risk such use may impose on their employers. These findings have important implications for corporate IT managers who must deal with this new workforce (of cherry pickers).

However, a new generation—born after 1998, or “Gen-Z”—will itself soon enter the workforce²³ and expected to behave differently—more concerned with risk and less on their own personal benefit.²⁷ This change in employees’ mindset will provide fruitful ground for further research. Further research should also look toward longitudinal datasets, as it would be interesting to find out how perceptions change before entering the workplace and then again after several months and years. Maybe IT managers could see cherry pickers evolve into socially minded corporate citizens over time. 

References

- Aldás-Manzano, J., Lassala-Navarre, C., Ruiz-Mafe, C., and Sanz-Blas, S. The role of consumer innovativeness and perceived risk in online banking usage. *International Journal of Bank Marketing* 27, 1 (2009), 53–75.
- Bauer, R.A. Consumer behavior as risk taking. In *Risk Taking and Information Handling in Consumer Behavior*, D.F. Cox, Ed. Harvard University Press, Cambridge, MA, 1967, 23–33.
- Breitsohl, H. and Ruhle, S. Differences in work-related attitudes between Millennials and Generation X: Evidence from Germany. In *Managing the New Workforce: International Perspective on the Millennial Generation*, E.S.W. Ng, S. Lyons, and L. Schweitzer, Eds. Edward Elgar, Cheltenham, U.K., 2014, 107–129.
- Cunningham, S.M. The major dimensions of perceived risk. In *Risk Taking and Information Handling in Consumer Behavior*, D.F. Cox, Ed. Harvard University Press, Cambridge, MA, 1967, 21–33.
- Derks, D., Duin, D., Tims, M., and Bakker, A.B. Smartphone use and work-home interference: The moderating role of social norms and employee work engagement. *Journal of Occupational and Organizational Psychology* 88, 1 (Mar. 2015), 155–177.
- Featherman, M.S. and Pavlou, P.A. Predicting e-services adoption: A perceived risk facets perspective. *International Journal of Human-Computer Studies* 59, 4 (2003), 451–474.
- Fishbein, M., Ed. *Readings in Attitude Theory and Measurement*. John Wiley & Sons, Inc., New York, 1967.
- Gewald, H. and Franke, J. The risks of business process outsourcing: A two-fold assessment in the German banking industry. *International Journal of Electronic Finance* 1, 4 (2007), 420–441.
- Hoehle, H., Zhang, X., and Venkatesh, V. An espoused cultural perspective to understand continued intention to use mobile applications: A four-country study of mobile social media application usability. *European Journal of Information Systems* 24, 3 (2015), 337–359.
- Hofstede, G. and Bond, M.H. The Confucius connection: From cultural roots to economic growth. *Organizational Dynamics* 16, 4 (Spring 1988), 5–21.
- Hofstede, G., Hofstede, G.J., and Minkov, M. *Cultures and Organizations: Software of the Mind*. McGraw Hill, New York, 2010.
- Kim, D. and Olufman, L. Determinants of corporate Web services adoption: A survey of companies in Korea. *Communications of the AIS* 29, 1 (2011), 1–24.
- Köffer, S., Anlauf, L., Ortbach, K., and Niehaves, B. The intensified blurring of boundaries between work and private life through IT consumerization. In *Proceedings of the European Conference on Information Systems* (Münster, Germany, 2015).
- Köffer, S., Ortbach, K., Junglas, I., Niehaves, B., and Harris, J. Innovation through BYOD? *Business & Information Systems Engineering* 57, 6 (Dec. 2015), 1–13.
- Kraut, R. and Burke, M. Internet use and psychological well-being: Effects of activity and audience. *Commun. ACM* 58, 12 (Dec. 2015), 94–100.
- Lee, M.-C. Factors influencing the adoption of Internet banking: An integration of TAM and TPB with perceived risk and perceived benefit. *Electronic Commerce Research and Applications* 8, 3 (May–June 2009), 130–141.
- Li, Y., Wang, X., Lin, X., and Hajli, M. Seeking and sharing health information on social media: A net-valence model and cross-cultural comparison. *Technological Forecasting and Social Change* (July 20, 2016); <https://doi.org/10.1016/j.techfore.2016.07.021>
- Liljander, V. and Strandvik, T. Estimating zones of tolerance in perceived service quality and perceived service value. *International Journal of Service Industry Management* 4, 2 (1993), 6–28.
- Liu, Y., Yang, Y., and Li, H. A unified risk-benefit analysis framework for investigating mobile payment adoption. In *Proceedings of the 2012 International Conference on Mobile Business*. AIS Electronic Library, Atlanta, GA, 2012; <http://aisel.aisnet.org/icmb2012/20>
- Lyons, S. and Kuron, L. Generational differences in the workplace: A review of the evidence and directions for future research. *Journal of Organizational Behavior* 35, S1 (Feb. 2014), 139–157.
- Miller, K.W., Voas, J., and Hurlburt, G.F. BYOD: Security and privacy considerations. *IT Professional* 14, 5 (Sept.–Oct. 2012), 53–55.
- Moore, G.C. and Benbasat, I. Development of an instrument to measure the perceptions of adopting an information technology innovation. *Information Systems Research* 2, 3 (1991), 192–222.
- Nadler, J.T., Morr, R., and Naumann, S. *Millennials, Media, and Research: Ageism and the Younger Worker*. Palgrave Macmillan, London, U.K., 2017.
- Ng, E.S.W., Schweitzer, L., and Lyons, S.T. New generation, great expectations: A field study of the millennial generation. *Journal of Business and Psychology* 25, 2 (June 2010), 281–292.
- Niehaves, B., Köffer, S., and Ortbach, K. IT consumerization: A theory and practice review. In *Proceedings of the 18th Americas Conference on Information Systems* (Seattle, WA, Aug. 9–11). Association for Information Systems, Atlanta, GA, 2012.
- Peter, J.P. and Tarpey Sr., L.X. A comparative analysis of three consumer decisions strategies. *Journal of Consumer Research* 2, 1 (June 1975), 29–37.
- Pew Research. *Teens, Social Media, and Privacy*. Pew Research, Washington, D.C., 2013; http://www.pewinternet.org/files/2013/05/PIP_TeensSocialMediaandPrivacy_PDF.pdf
- Rapoza, J. *The increasing costs of BYOD*. Aberdeen Group Market Alert, Boston, MA, 2014.
- Rogers, E.M. *Diffusion of Innovations*. Free Press, New York, 1995.
- Srite, M. and Karahanna, E. The role of espoused national cultural values in technology acceptance. *MIS Quarterly* 30, 3 (Sept. 2006), 679–704.
- Steelman, Z.R., Lacity, M., and Sabherwal, R. Charting your organization’s bring-your-own-device voyage. *MIS Quarterly Executive* 15, 2 (2016), 85–104.
- Twenge, J.M. The evidence for Generation Me and against Generation We. *Emerging Adulthood* 1, 1 (2013), 11–16.
- Venkatesh, V., Morris, M., Davis, G., and Davis, F. User acceptance of information technology: Toward a unified view. *MIS Quarterly* 27, 3 (Sept. 2003), 425–478.
- Verizon. *Data Breach Investigations Report, 2015*; https://iapp.org/media/pdf/resource_center/Verizon_data-breach-investigation-report-2015.pdf
- Vodanovich, S., Sundaram, D., and Myers, M. Digital natives and ubiquitous information systems. *Information Systems Research* 21, 4 (2010), 711–723.
- Weegee, A. and Gewald, H. Factors influencing future employees’ decision making to participate in a BYOD program: Does risk matter? In *Proceedings of the 22nd European Conference on Information Systems* (Tel Aviv, Israel), 2014.
- Weegee, A., Wang, X., and Gewald, H. IT consumerization: BYOD-program acceptance and its impact on employer attractiveness. *Journal of Computer Information Systems* 56, 1 (Fall 2016), 1–10.
- Weiß, F. and Leimeister, J.M. Consumerization: IT innovations from the consumer market as a challenge for corporate IT. *Business & Information Systems Engineering* 54, 6 (Dec. 2012), 363–366.

Heiko Gewald (heiko.gewald@hs-neu-ulm.de) is a research professor of information management and director of the Center for Research on Service Sciences at Neu-Ulm University of Applied Sciences, Neu-Ulm, Germany.

Xuequn Wang (xuequnwang1600@gmail.com) is a lecturer at Murdoch University School of Engineering and Information Technology, Perth, Australia.

Andy Weegee (Andy.Weegee@hs-neu-ulm.de) is a research assistant at Neu-Ulm University of Applied Sciences, Neu-Ulm, Germany, and a Ph.D. candidate in the University of Bamberg, Bamberg, Germany.

Mahesh S. Raisinghani (mraisinghani@twu.edu) is a professor of management information systems in the MBA (Executive Track) in the College of Business Administration at Texas Woman’s University, Denton, TX.

Gerald Grant (Gerald.Grant@carleton.ca) is a director of the Centre for Information Technology, Organizations, and People and associate professor of information systems in the Sprott School of Business, Carleton University, Ottawa, Canada.

Otávio P. Sanchez (otavio.sanchez@fgv.br) is a professor in the Business Administration Ph.D. Program at Fundação Getúlio Vargas, São Paulo, Brazil.

Siddhi Pittayachawan (siddhi.pittayachawan@rmit.edu.au) is a senior lecturer of information systems and supply chain management in the School of Business IT and Logistics at RMIT University, Melbourne, Australia.

Copyright held by the authors.
Publication rights licensed to ACM. \$15.00

Exploring the technical and ethical issues surrounding Internet advertising and ad blocking.

BY STEPHEN B. WICKER AND KOLBEINN KARLSSON

Internet Advertising: Technology, Ethics, and a Serious Difference of Opinion

“Every time you block an ad, what you’re really blocking is food from entering a child’s mouth.”²⁵

“In reality, ad blockers are one of the few tools that we as users have if we want to push back against the perverse design logic that has cannibalized the soul of the Web.”²⁶

IN FALL 2015, Apple introduced a “content blocking” extension point into its Safari mobile browser, providing a hook for software that prevents advertisements from being loaded when Web pages

are rendered.²⁷ As it turned out, large numbers of people wanted to do just that.²⁸ Ad blockers had been available for some time, but their potential use in the world’s most popular mobile browser heightened their saliency and brought the debate over their use—a debate sometimes serious and nuanced, but often frivolous—into the mainstream media.²⁹

To put the issue into perspective, consider the following provided by PageFair, “a leading provider of counter ad block solutions to Web publishers,” in its 2015³⁰ and 2016³¹ reports on ad blocking:

- ▶ Ad blocking was estimated to have cost publishers nearly \$22 billion during 2015.

- ▶ As of November 2016, at least 309 million people are blocking advertising on their smartphones.

- ▶ 298 million of these people use an ad blocking browser, more than twice the number using blocking browsers in 2015.

- ▶ Ad blocking is particularly popular in emerging markets, with the largest number of active monthly users in China, India, and Indonesia. The U.S. is in ninth place.

In its 2016 report, PageFair made the following prediction: *Mobile ad blocking is a serious threat to the future of media and journalism in emerging markets, where people are coming online for the first time via relatively expensive*

» key insights

- Internet advertisers use networks of supply- and demand-side platforms and automated auctions to deliver targeted advertising to readers in a matter of milliseconds.
- Ad blocking is an existential threat to the Internet advertising industry, with costs to advertisers ranging in the tens of billions of dollars.
- The argument that ad blockers violate an implicit contract between the reader and content provider fails on legal grounds.
- A virtue ethics-based analysis clearly supports ad blockers, while also pointing to solutions that may benefit all stakeholders.



or slow mobile connections. Usage in Western economies is likely to grow as more manufacturers and browsers start to include ad blocking as a feature.³¹

Given the amount of money involved in advertising, one might expect a certain amount of invective on the subject of ad blocking. One would be correct. Ad blocking has been referred to as “evil” and as a form of “theft.”³² *Ad Age*, an advertising industry trade magazine, accused ad blockers of being exploitative, extortionate, and anti-democratic, all within the space of a single sentence:

*As abetted by for-profit technology companies, ad blocking is robbery, plain and simple—an extortionist scheme that exploits consumer disaffection and risks distorting the economics of democratic capitalism.*³³

Randall Rothenberg, president and chief executive officer for the Interactive Advertising Bureau accuses ad blocking “profiteers” of “stealing from publishers, subverting freedom of the press, operating a business model predicated on censorship of content, and ultimately forcing consumers to pay more money for less—and less diverse—information.”³⁴

On the other side of the debate, many have pointed to the ads themselves as fostering needless consumption while being tasteless, intrusive, and evil (this word occurs a lot in these discussions), while suggesting that the advertising industry brought ad blocking upon itself.¹

There are purely technical issues as well. The technology that allows Internet advertisers to better target potential consumers slows the loading of Web pages and places a significant burden

on wireless cellular links, a burden that is usually funded by unwilling users. The ad-blocking software provider Shine, an Israeli startup that began life in 2011 as an anti-virus software developer, estimates that advertising consumes between 10% and 50% of user data plans, depending on user location. A typical mobile gaming app with advertising was found to consume 5Mb over a five-minute session, but only 50Kb with ad blocking in place.¹⁷

Shine produces ad-blocking software that can be incorporated into cellular datacenters. In June 2016, the U.K. cellular service provider Three became the first to conduct trials using this software to block ads on cellular data connections.¹² Given that marketers are expected to have spent over \$100 billion on mobile ads in 2016,¹⁰ the response is expected to be extreme.

In this article, we explore how advertising networks and ad blockers work. We further consider how ad blockers are subverted, and whether they are ethical. The ethical analysis yields mixed results, but it does, however, suggest a solution that empowers users, allowing them to select the types of ads that they see and how often they see them.

The Technology of Ad Networks and Ad Blockers

Web browsers request a Web page from a server by sending an HTTP GET command to the appropriate Internet host. The host responds with HTML code that the Web browser uses to render the desired page and present it to the user. This much is both simple and ubiquitous, but the details, particularly when advertising is involved, are much more complicated.

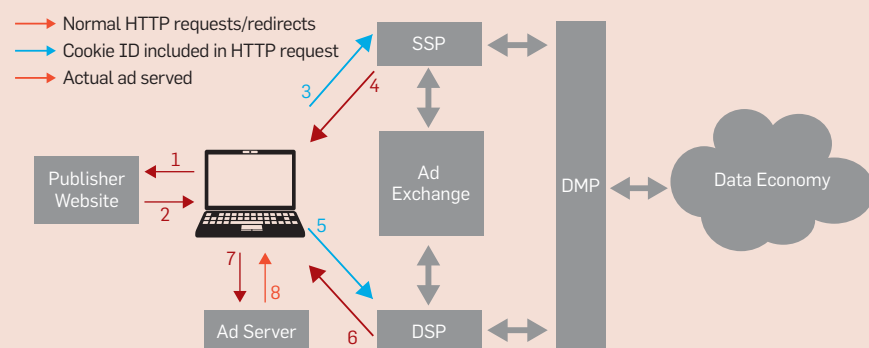
Suppose the Web browser requests a page from a content publisher that supports his or her work through advertising (this is represented in the accompanying figure by link 1). Most publishers do not generate their own advertising content, so they will embed requests for advertising into the HTML files they send to requesting users (link 2). When the requesting host attempts to render the HTML file, it will generate requests for advertisements from an ad exchange. The ad exchange, as shown in the figure, sits at the center of a network consisting of supply side and demand side entities. The supply side entities provide information about the user, while the demand side entities provide advertising in response to requests from publishers.

The HTML code provided by the publisher directs the host to a supply-side platform (SSP—link 3). The request sent to the SSP includes a cookie—a small string of information that was previously stored by the SSP on the user’s computer. The cookie enables the SSP to craft a response that is specifically tailored to the requesting user. In this case, the cookie will include a user ID that the ad exchange can use to coordinate bidding for an advertisement.

The ad exchange forwards the user ID and any other information that it may have about the requesting user to one or more demand-side platforms (DSPs) that place bids on behalf of advertisers for the opportunity to display their ads. Through a process known as cookie syncing, the DSPs are able to match the SSP cookie ID to a user profile, which is often stored and managed by a separate entity called a data management platform (DMP).

As multiple SSPs and DSPs can use the same DMP, the DMP may link a wide range of user IDs to the same person. This enables all interested parties (other than the user) to exchange information on the user and form a more complete picture of that user’s browsing history. First-party websites may also participate in the process, providing yet more user information. For example, if a user supplies an email address to a website to sign up for its newsletter, the email address can be linked at the DMP to the cookie IDs associated with that user. If the user provides a name and address to a website, that information may also

Internet ad delivery is a complex process involving multiple redirections, synchronization of user information, and an auction, all in a few tens of milliseconds.



be linked to the cookie IDs. The DMP may take this a step further by including information inferred from the user's social media activity, purchase history on various sites, search history, and email messages. Finally, the DMP may have access to data gathered offline. Data aggregators are known to collect data from publicly available records, including licensing records (for example, licenses for doctors, lawyers, pilots, or hunters or fishermen licenses), voter registration databases, court records, and DMV records, as well as buying data from commercial sources including brick-and-mortar store purchase histories and transaction information from financial services companies.⁹ Data aggregators also buy and sell information from each other. This whole system of transactions is often referred to as “the data economy.” Through this data economy, the DMP is able to build a strikingly detailed simulacrum of an individual consumer, a simulacrum whose accuracy drives the advertisers' return on investment, and whose inaccuracy may drive the consumer to distraction.

If a DSP determines the user profile fits its target audience, it places a bid for advertising space on the web page being rendered by the host computer. The ad exchange selects an ad from among the bidding DSPs; a Vickrey auction is generally used, where the highest offer is selected and the amount paid is that offered by the second highest bidder. The winning DSP provides a URL for retrieving the ad. In an “impression”-based system, an agency ad server determines whether the ad is actually downloaded, and pays the publisher accordingly, with the cost per thousand impressions (CPM) being the most widely used statistic in Internet advertising.³⁵ All of this happens within tens of milliseconds, though the actual loading of the winning ad into the user's browser may take far longer depending on the bandwidth of the user connection and the size of the ad.

As a result of this process, the publisher of the content often has limited control over the safety, quality, or tastefulness of the ad seen by the content consumer. A publisher may, for example, be able to prevent advertisements from a particular advertiser or class of advertisers, but she may not be able to exercise finer control.



Given the amount of money involved in advertising, one might expect a certain amount of invective on the subject of ad blocking. One would be correct.



Ad blockers can use several methods to disrupt the process described earlier, thus prevent ads from being displayed. Many prominent ad blockers, such as Adblock Plus and its variants, block ads by preventing the browser from sending HTTP requests to certain URLs. The URL blacklist for a given blocker is often a crowd sourced effort, such as EasyList, the default blacklist used in Adblock Plus. EasyList is probably the most widely used blacklist; the number of EasyList downloads was used by PageFair and Adobe to estimate the prevalence of ad blocking in their 2015 joint report.³⁰

While URL blacklisting appears to be the most common method of ad blocking, the Electronic Frontier Foundation's Privacy Badger takes a different approach,³⁶ attempting to learn which domains and sites are tracking a user and blocking the ones that do. It detects behavior such as the use of uniquely identifying cookies, canvas fingerprinting, and the appearance of the same third-party site at multiple domains. As such, it blocks very few domains at first, but the more it is used, the more it learns to block. It should be noted that Privacy Badger aims to prevent tracking, not ads; but since the two are intimately connected, it often serves both purposes.

A third ad blocking method blocks website elements fitting certain patterns; for example, it could look for the “iframe” HTML tag and check to see if it contains text strings like “Sponsored” or links to a URL with the word “ad” in it. This method can block advertising served by the Web site itself as opposed to just third parties, advertising that includes ads embedded in search results and social media feeds.

This content filtering can happen at the client or at an intermediate proxy. Some ad blockers use a root certificate to redirect browser requests to a VPN or proxy that removes ad content using the methods previously mentioned before forwarding the HTML code to the browser. This approach can block ads for mobile apps as well as browsers, but it comes with the risks associated with having third parties interfere with browser traffic, risks that include the classic man-in-the-middle attacks. Apple recently removed several ad-blocking apps from its app store on this basis.³⁷

Publishers will sometimes try to circumvent attempts at ad blocking. Anti-ad blocking usually works by serving a fake ad in some way and verifying that it has been loaded or displayed. If it fails to load, the site stops displaying the primary content or refuses to load it in the first place. For example, a site can contain an iframe ostentatiously marked as “Advertising” and then use JavaScript to see if it was displayed.^a If the iframe is not displayed, the site does not provide the primary content. Similarly, the browser can be directed to load a JavaScript with a name, such as “ads.js,” that can be found in common ad blocker filter rules and check to see if it is run. Aside from trying to explicitly detect ad blockers, ad networks can obfuscate the URLs of their ads, such as by using IP addresses instead of domain names.

Ad blockers can often adapt, circumventing new anti-ad blocking mechanisms. Facebook recently announced it would prevent ad blocking,³⁸ only to have Adblock Plus announce a few days later that it found a way to defeat Facebook’s prevention technique.³⁹ This is but one example of the evolving arms race between publishers and ad blockers.¹⁶

Though the initial motivation for ad blocking may be annoying ads or tracking, increased computer security is a major side benefit. Online ads are usually pieces of code as opposed to static images or text. The end result of the ad auction process described earlier is that the user’s browser is redirected to a URL of the advertiser’s choosing. The retrieved object may take the form of JavaScript, Flash, or even Java code. Vulnerabilities in these frameworks can be used to execute malicious code on the client machine without the user noticing anything out of the ordinary. Even though browser support for Java and Adobe Flash is being phased out,⁴⁰ vulnerabilities in these frameworks are still being exploited. Java exploits are on the decline, but Flash vulnerabilities are still some of the most common vehicles for malvertising.⁴ While ad networks have measures in place to detect malvertising, there are ways to circumvent and avoid detection, at least temporarily, such as serving a legitimate ad until the ad network has approved



Publishers will sometimes try to circumvent attempts at ad blocking. Anti-ad blocking usually works by serving a fake ad in some way and verifying it has been loaded and displayed.



the ad, only serving malvertising every 10th or 20th time, and not serving malvertising to certain IP addresses.⁵ Even large and reputable websites have been known to accidentally serve malvertising, making malvertising a potential problem for every Internet user.

Is Ad Blocking a Breach of Contract?

It would take a long law review article, and one written by another set of authors, to properly address the legality of ad blocking. We do, however, wish to address the oft-cited argument that the provision of free content that contains ads is done under an “implicit contract.”⁴¹ Under this contract, the consumer is provided with free content in return for the user’s agreement to view advertisements. This is not a new argument, as it has been applied by network executives to broadcast television for many years, sometimes in a very extreme form. In 2002, Jamie Kellner, then CEO of Turner Broadcasting, suggested that any systematic practice of using the bathroom during commercials was stealing.⁷

In the U.S., the legal concept underlying this argument is the “implied-in-fact” contract.^b The law is summarized as follows:

To establish the existence of an implied in fact contract, it is necessary to show: an unambiguous offer, unambiguous acceptance, mutual intent to be bound, and consideration. However, these elements may be established by the conduct of the parties rather than through express written or oral agreements.⁴²

As an example, suppose you agree to wash your neighbor’s car once a week. You receive payment for each of the first six weeks, but upon washing his car the seventh time, your neighbor refuses to pay because there was no written agreement. Most courts would agree that there was an implied-in-fact contract as evidenced by the conduct of the parties for the first six weeks. Your neighbor has to pay.

Now consider a real example in which it was found that there was no

a See, for example, <http://adblockingdetector.johnmorris.me/how-thisplugin-works/>.

b U.S. contract law allows for two other types of contracts: express contract (written) and implied-in-law contracts (also called “quasi contracts,” they are more legal obligations than true contracts). We only address U.S. contract law here; other jurisdictions may be substantially different, and are well beyond our expertise.

implied-in-fact contract. In 1917, the U.S. leased a pier from the Baltimore and Ohio railroad for the purpose of handling supplies destined for the war in Europe. An earlier fire was believed to have been an act of sabotage, so soldiers were deployed to guard the pier and surrounding equipment. The weather was cold, and the troop commander often complained about the tents in which his men were forced to live. A railroad official offered to build temporary barracks. Though there was never any discussion of compensation, the barracks were built. The railroad later sued to recover the cost of the construction, arguing there had been an implied-in-fact contract. In what became the 1923 case of *Baltimore & Ohio R. Co. v. United States*,² the Supreme Court disagreed. The Court stated that an “implied agreement” required “a meeting of minds inferred, as a fact, from conduct of the parties in the light of surrounding circumstances.” The Court found there had been no such meeting of minds, as the railroad company never intimated that it would expect payment from the government.

It follows that there are several reasons the alleged quid pro quo of viewing ads in return for free Internet content fails to rise to the level of an implied-in-fact contract. First, as with the *Baltimore & Ohio* case, there was no unambiguous offer. The Internet content consumer is rarely told precisely what is going to be loaded into his or her Web browser, and what is expected in return. Content consumers suffer the embedding of ads and, on occasion, trackers and other forms of spyware into their Web browsers without receiving any notice from the content provider whatsoever. In fact, as we have seen, the content provider may not know what is being injected into the consumer’s browser.

Second, the alleged agreement fails to satisfy the unambiguous acceptance element. Unlike the lawn-mowing example, there is no prior conduct that indicates a general understanding that an agreement is in place. The popularity of ad blockers^{20,30,31} indicates that most consumers do not want to see the ads, and clearly have not agreed to do so. A Reuters survey provides further evidence, indicating that even those who do not employ ad blockers are ignoring or avoiding the ads:

*More generally, a third or more (39% in the U.K. and 30% in the U.S.) say they ignore ads. Around three in 10 (31%/29%) say they actively avoid sites where ads interfere with the content.*²⁰

Are Ad Blockers Unethical?

In *After Virtue*, Alasdair MacIntyre describes the breakdown in ethical argument that occurs when the foundations for ethical systems are cut away, leaving proponents of differing perspectives to argue past each other without any basis for decisive engagement.¹⁴ Ad blocking provides a canonical example, as we have one group arguing for individual rights (the right to receive payment for one’s effort in providing content), while the other group argues for the general welfare (an Internet devoid of continual distraction caused by tasteless advertising). It is not clear how the two arguments can be reconciled, or how one can clearly overcome the other. We suggest a solution lies in a technologically mediated meeting of minds, but before we consider the solution, we offer a more detailed account of the ethical arguments.

The utilitarian approach, first propounded by Jeremy Bentham and John Stuart Mill in the late 18th and early 19th centuries, is based on the familiar precept that “it is the greatest happiness of the greatest number that is the measure of right and wrong.”³ In what follows, we will consider *act utilitarianism*, which focuses on the consequences of individual actions. We will also substitute “well-being” for “happiness” to counter some of the more obvious criticisms of utilitarianism.

Does the use of ad blockers create the greatest well-being for the greatest number? Those affected by the decision to block ads include the following:

- ▶ Ad blocking users,
- ▶ Ad viewing users,
- ▶ Content generators,
- ▶ Content publishers, and
- ▶ Advertisers

Should users choose to employ ad blockers, the following will arguably result:

- ▶ The ad blocking users will see fewer advertisements.
- ▶ The content generators will receive less revenue per reading user.
- ▶ The content publishers will receive less revenue per reading user.

▶ Advertisers will seek other venues for their advertising dollars.

▶ Some content generators will stop generating content.

▶ Some content publishers will stop publishing content.

▶ Some content publishers will publish content of lower quality.

▶ There will be less free content available to all users on the Internet, and the content that remains freely available will, in some cases, be of reduced quality.

It is important to provide some context for the suggestion that the quality of online content will be diminished by a general acceptance of ad blocking. Newspaper journalism was in decline well before the advent of ad blocking, or even the advent of the Internet, primarily because of the failure of its core business model.¹⁵ The business model was that of a quasi-monopoly: competition was limited, so that a local paper could charge higher prices for advertising, and then use the revenue to maintain reporters across the world. In essence, the local Wal-Mart paid for the Baghdad bureau through its advertising dollars. The limit on competition was due to a fact of technology: printing presses were very expensive to operate and maintain, so all but the largest municipalities could only sustain one or two (print) newspapers at any given point in time.²² In a pre-Internet world, the papers acted as an intermediary between advertisers and consumers, charging both for the opportunity to communicate. In a multi-newspaper market, the equilibrium was often unstable; a notable scoop could send more advertising dollars to the scooping paper, allowing the scooper to grow (literally) fatter and more attractive to the buyers.

The unraveling of this relationship began with the television era and the movement of affluent readers from the inner city to the suburbs. National and retail advertisers moved their dollars to television, and newspapers came to depend more on classified ads.⁶ With the advent of the Internet in general and Craig’s List in particular (founded in 1995), classified advertising revenue also began to leave the newspapers’ balance sheets. By 2010 the newspaper industry was in deep decline, with many major players facing bankruptcy (for example, the Tribune Company in

2008), and others left to cope with dramatically reduced staffs.

The consequences of a general use of ad blockers may thus be characterized as a *further* reduction in the quality of free online content through the departure of some Internet content generators and publishers to other ways of making a living. For large numbers of consumers these are apparently acceptable outcomes given what they avoid: the problems associated with spyware and the relentless distraction of advertising. There is also evidence that Internet readers do not greatly value what they are reading; given the choice between paying for the content and losing it, most prefer the latter. The aforementioned Reuters survey found that only 10% of online users appeared to be willing to pay for once-free news content.

*After a sharp upturn in 2012–2013—when a large number of paywalls were introduced—our data shows very little change in the absolute number of people paying for digital news over the past year. In most countries the number paying for any news is hovering around 10% of online users and in some cases less than that.*²⁰

If Internet readers and users of ad blockers are rational actors who are making decisions based on their individual well-being, and as the readers outnumber the writers and advertisers, one may conclude that the use of ad blockers provides the greatest well being for the greatest number. From a utilitarian perspective, ad blocking is ethical; the content providers should look for a better business model.

The counterargument is ready at hand: this analysis clearly does not take into account all stakeholders; the content generators and publishers, for example, would almost certainly not be pleased with the consequences of this utilitarian calculus. This is an example of a key criticism of utilitarianism; namely, that in emphasizing aggregate well being, some individuals may be left in far worse condition than before.

A deontic analysis avoids this particular problem. Immanuel Kant suggested in his *Groundwork of the Metaphysic of Morals* that there is a single primary moral obligation, which he referred to as the “categorical imperative” (CI). Kant offered several formulations of CI, including one that sounds very much like the golden rule: “act so that

you use humanity, as much in your own person as in the person of every other, always at the same time as end and never merely as means.”¹³ Ad blocking readers arguably do not satisfy this formulation—they treat the content generators as means rather than an end in themselves, taking their work product without respecting their efforts to make a living. It appears that Kant is on the side of the advertisers, while Bentham favors the general reader.

Contractualism, an ethical theory related to Kant’s deontological approach,¹⁸ more clearly takes into account all interested parties, while pointing to a potential solution. In *What We Owe Each Other*, T.M. Scanlon²¹ offers the following ethical rule for action (emphasis added):

An act is wrong if its performance under the circumstances would be disallowed by any set of principles for the general regulation of behavior that no one could reasonably reject as a basis for informed, unforced, general agreement.

In establishing rules for behavior, Scanlon suggests that we must consider the perspectives of all stakeholders, and define a basis for informed general agreement. This would require communication between all stakeholders, something that is sorely lacking in the context of online advertising. We will return to this point when we consider possible solutions.

The third and final approach to be considered shifts the balance of the argument in favor of the general reader, but on a far firmer basis than the arguments of Bentham et al. Aretaic, or *virtue* ethics, emphasizes virtues of mind and character.^{8,11} Virtue ethics originated with Aristotle’s *Nicomachean Ethics* and his notion that the ultimate aim (*telos*) of an individual is to live a virtuous life. A virtuous life is a life lived according to reason, where decisions are based on a set of values held dear by the individual. Virtue ethics thus involve the questions of “what is desirable, good or morally worthwhile in life?” “What values should we pursue for ourselves and others?”⁸

Virtue ethics has enjoyed a recent resurgence, both in philosophy departments and in schools of technology. With regard to the latter, value-based design practices have been developed based on various lists of fundamental

human values. For example, in *Ethical IT Innovation*, Sarah Spiekermann points to both Aristotle and Maslow while concluding that technical design must be based on an understanding that knowledge, freedom, and autonomy are preconditions for human growth, self-esteem, friendship and self-actualization.²⁴

At best, the design of advertising technology shows little concern for knowledge, freedom, and autonomy of consumers. At worst, advertising technology actively works to subvert these values. This subversion can be seen through the lens of the “attention economy,” a term coined by Herbert Simon to capture the finite nature of the individuals’ attention in the face of a seemingly infinite amount of information.²³ The attention economy is reflected in advertisers’ insertion of themselves into virtually all personal interactions in everyday life, ranging from highway billboards to doctors’ offices to the bottoms of the trays at airport security. Writing for the “Practical Ethics” blog of Oxford University, James Williams argues the resulting distractions are more than an annoyance, they “keep us from living the lives we want to live.”

*In the short term, distractions can keep us from doing the things we want to do. In the longer term, however, they can accumulate and keep us from living the lives we want to live, or, even worse, undermine our capacities for reflection and self-regulation, making it harder, in the words of Harry Frankfurt, to “want what we want to want.” Thus there are deep ethical implications lurking here for freedom, well-being, and even the integrity of the self.*¹⁹

From a virtue ethics standpoint, it follows that the design of Internet advertising technology is itself unethical in that it works against the human project of self-creation. Ad blockers are thus not only ethical, but are literally a matter of self-defense. Quoting the Practical Ethics blog once again:

*In reality, ad blockers are one of the few tools that we as users have if we want to push back against the perverse design logic that has cannibalized the soul of the Web.*⁶

Solutions and Conclusion

The advertising delivery systems described in this article are the antithesis of value-based design. The values that Spiekermann and others point to as a foundation for virtue-based

design—knowledge, freedom, and autonomy—are precisely the values that online advertising systems most systematically undermine.

Internet advertisers exchange information about users without their knowledge or control, using that information to manipulate users into behavior they might not otherwise have exhibited. This summary may seem harsh, and some may argue that advertisers would happily engage in more ethical behavior if better channels of communication were provided to interested consumers. A solution beneficial to all may lie in a virtue-based redesign. Such a redesign would embed T.M. Scanlon's suggestion that there be an "informed, unforced, general agreement" among all parties. The agreement would be based on a system that provides revenue for content generators and connects advertisers to interested consumers while reducing the deleterious impact of the current system of advertising on the reading public. The key step lies in empowering the reading/consuming public—letting them choose whether they will download ads, and if so, what type of ads. Should a reader choose not to download ads, he or she should be given the opportunity to pay for ad-free content.

The supporting technology for such a system already exists in the current ad networks. Recall the current scheme of directed Internet advertising relies on the use of cookies stored on user machines. These cookies are sent to service-side and demand-side platforms to obtain directed advertising for insertion into content initially requested by a user. Suppose the cookies are replaced by information explicitly provided by the user that indicates buying habits and interest in specific consumer goods. The data management platform (DMP) would request, coordinate, and update this user-supplied information as necessary. Rather than inferring potential sales from browsing habits, advertising networks could make advertising bidding decisions based on the clearly expressed desires of potential consumers. Such a system would increase the agency of the browsing user, while potentially increasing return on investment for advertisers.

Such a solution will require careful design and far more communication between stakeholders than currently takes place, but it offers the potential for clearly informing readers of their options, options upon which they can exercise rational choice in pursuit of their own individual goals. We hope that advertisers see this as an opportunity.

We have argued that ad blocking is not a violation of an existing contract (at least in U.S. law). This does not mean that ad blocking is beyond the reach of earnest lobbyists and subsequent legislation. One might expect, however, that such legislation would not be very popular with the general public. We hope the agreement suggested here takes form before the battle between advertisers and ad blockers escalates any further.

Acknowledgments

The authors greatly acknowledge the assistance of Sarah Wicker and Adam Engst with several technical, legal, and ethical issues. We also thank the editor and reviewers for their time, effort, and expertise. 

References

- Alexander, J., Crompton, T. and Shrubsole, G. Think Of Me As Evil? Opening The Ethical Debates In Advertising. Public Interest Research Centre (PIRC) and WWF-UK, Oct. 2011.
- Baltimore & Ohio R. Co. v. United States* 261 U.S. 592 (1923).
- Bentham, J. A fragment on government. (1776). Reprinted in *The Collected Works of Jeremy Bentham*. J.H. Burns and H.L.A. Hart, eds. The Athlone Press, London, U.K. 1977, 393.
- Cisco. Cisco 2016 Midyear Cybersecurity Report; <http://www.cisco.com/c/en/us/offers/sc04/2016-midyearcybersecurity-report/index.html>.
- Cyphort Labs. The Rise of Malvertising, 2015; <http://go.cyphort.com/Malvertising-Report-15-Page.html>.
- Downie, Jr., L. and Schudson, M. The reconstruction of American journalism. *Will the Last Reporter Please Turn out the Lights: The Collapse of Journalism and What Can Be Done To Fix It*. R.W. McChesney and V. Pickard, eds. The New Press, NY, 2011.
- EFFector* 15, 15 (May 24, 2002); <https://www.eff.org/effector/15/15>
- Frankena, W. *Ethics* (2nd Edition). Prentice Hall, Upper Saddle River, NJ, 1973.
- FTC. Data Brokers: A Call for Transparency and Accountability; <https://www.ftc.gov/system/files/documents/reports/data-brokerscall-transparency-accountability-report-federal-trade-commissionmay-2014/140527databrokerreport.pdf>
- Hof, R. Mobile ads will smash \$100 billion mark worldwide in 2016. *Forbes* (Apr. 2, 2015); <http://www.forbes.com/sites/roberthof/2015/04/02/mobile-adswill-smash-100-billion-mark-worldwide-in-2016>
- Hursthouse, R. *On Virtue Ethics*. Oxford University Press, 2002.
- Jackson, J. Three network to run 24-hour ad blocking trial. *The Guardian* (May 25, 2016); <https://www.theguardian.com/media/2016/may/26/three-network-to-run-24-hour-ad-blocking-trial>
- Kant, I. *Groundwork for the Metaphysics of Morals*. A. Wood, ed. Yale University Press, 2002, 37.
- MacIntyre, A. *After Virtue: A Study in Moral Theory* (3rd Edition). University of Notre Dame Press, South Bend, IN, 2007.
- McChesney, R.W. and Pickard, V. Eds. *Will the Last Reporter Please Turn out the Lights: The Collapse of Journalism and What Can Be Done To Fix It*. The New Press, NY, 2011.
- Nithyanand, R. et al. Adblocking and Counter-Blocking: A Slice of the Arms Race (2016); arXiv:1605.05077.
- O'Reilly, L. This ad blocking company has the potential to tear a hole right through the mobile Web—and it has the support of carriers. *BusinessInsider.com*, May 13, 2015.
- Parfit, D. *On What Matter 1* (2011), 412–413. Oxford University Press.
- Practical Ethics Blog. University of Oxford; <http://blog.practicaethics.ox.ac.uk/2015/10/why-its-ok-to-blockad>
- Reuters Institute Digital News Report 2015. Reuters Institute, University of Oxford, https://reutersinstitute.politics.ox.ac.uk/sites/default/files/Reuters%20Institute%20Digital%20News%20Report%202015_Full%20Report.pdf
- Scanlon, T.M. *What We Owe Each Other*. Belknap Press, Cambridge, MA, 2000.
- Shirky, C. Newspapers and thinking the unthinkable. *Will the Last Reporter Please Turn out the Lights: The Collapse of Journalism and What Can Be Done To Fix It*. R.W. McChesney and V. Pickard, Eds. The New Press, NY, 2011.
- Simon, H.A. Designing organizations for an information-rich world. *Computers, Communications, and the Public Interest*. Johns Hopkins University Press, Baltimore, MD, 1971.
- Spiekermann, S. *Ethical IT Innovation: A Value-Based System Design Approach*. CRC Press, Boca Raton, FL, 2015.
- <http://www.tomsguide.com/us/ad-blocking-is-stealing-news-20962.html>
- <http://blog.practicaethics.ox.ac.uk/2015/10/why-its-ok-to-block-ads/>
- <https://developer.apple.com/library/ios/releasenotes/General/WhatsNewInIOS/Articles/iOS9.html>
- <https://adblockplus.org/blog/adblock-plus-for-ios-9-finally-here-and-pssst-it-s-free>
- <http://time.com/4052033/apple-iphone-ios-9-ad-blockers/>
- <https://blog.pagefair.com/2015/ad-blocking-report/>
- <https://pagefair.com/downloads/2016/05/Adblocking-Goes-Mobile.pdf>
- <http://fortune.com/2015/09/18/dear-apple-i-may-rob-yourstore/> <http://www.tomsguide.com/us/ad-blocking-isstealing-news-20962.html>
- <http://adage.com/article/digitalnext/ad-blocking-unnecessaryinternet-apocalypse/300470/>
- <http://www.iab.com/news/rothenberg-says-ad-blocking-is-awar-against-diversity-and-freedom-of-expression/>
- <http://www.allbusiness.com/web-advertising-and-cpm-aquick-guide-for-small-businesses-2646-1.html>
- <https://www.eff.org/privacybadger>
- <http://www.theguardian.com/technology/2015/oct/09/applemoves-iphone-adblockers-facebook-third-party-apps>
- <http://www.wsj.com/articles/facebook-will-force-advertising-on-ad-blocking-users-1470751204>
- <http://www.cnet.com/news/facebook-adblock-plus-workaround-august-11>
- <http://www.extremetech.com/computing/209888-mozilla-firefox-kills-flash-by-default-security-chiefs-calls-for-adobe-to-issue-an-end-of-life-date>, <http://www.bbc.com/news/technology-36301904>
- <http://www.computerworld.com/article/2487367/e-commerce/ad-blockers-a-solution-or-a-problem.html>
- https://www.law.cornell.edu/wex/contract_implied_in_fact

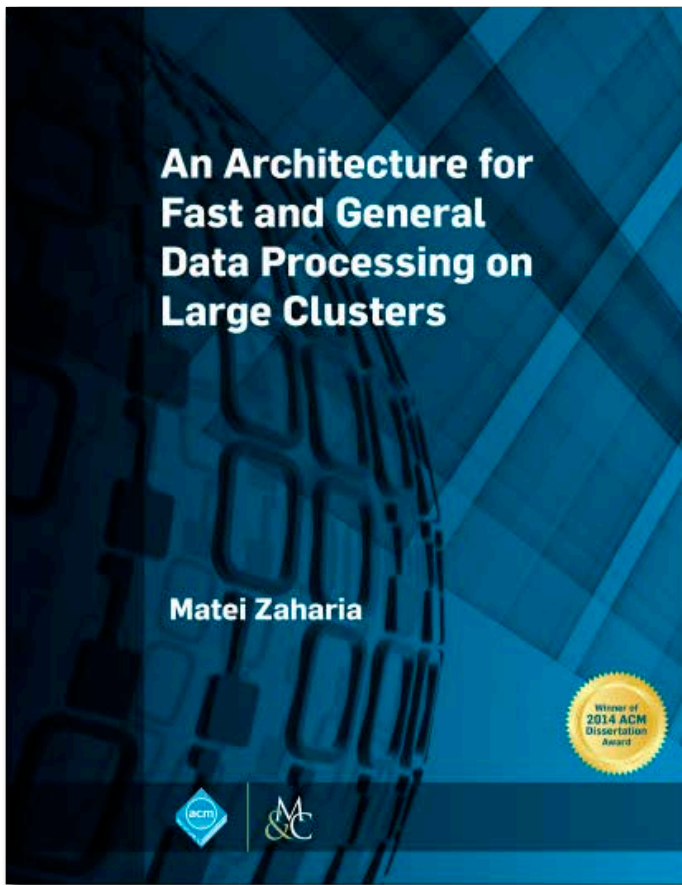
Stephen B. Wicker (wicker@ece.cornell.edu) is a professor in the School of Electrical and Computer Engineering at Cornell University, Ithaca, NY.

Kolbeinn Karlsson (kk752@cornell.edu) is a graduate student in the School of Electrical and Computer Engineering at Cornell University, Ithaca, NY.

© 2017 ACM 0001-0782/17/10 \$15.00



Watch the authors discuss their work in this exclusive *Communications* video. <https://cacm.acm.org/videos/internet-advertising>



**Proposing an
architecture for cluster
computing systems
that can tackle
emerging data
processing workloads
at scale.**

This book, a revised version of the 2014 ACM Dissertation Award winning dissertation, proposes an architecture for cluster computing systems that can tackle emerging data processing workloads at scale. Whereas early cluster computing systems, like MapReduce, handled batch processing, our architecture also enables streaming and interactive queries, while keeping MapReduce's scalability and fault tolerance. And whereas most deployed systems only support simple one-pass computations (e.g., SQL queries), ours also extends to the multi-pass algorithms required for complex analytics like machine learning. Finally, unlike the specialized systems proposed for some of these workloads, our architecture allows these computations to be combined, enabling rich new applications that intermix, for example, streaming and batch processing.

We achieve these results through a simple extension to MapReduce that adds primitives for data sharing, called Resilient Distributed Datasets (RDDs). We show that this is enough to capture a wide range of workloads. We implement RDDs in the open source Spark system, which we evaluate using synthetic and real workloads. Spark matches or exceeds the performance of specialized systems in many domains, while offering stronger fault tolerance properties and allowing these workloads to be combined. Finally, we examine the generality of RDDs from both a theoretical modeling perspective and a systems perspective.

This version of the dissertation makes corrections throughout the text and adds a new section on the evolution of Apache Spark in industry since 2014. In addition, editing, formatting, and links for the references have been added.



<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/clusters>

research highlights

P. 80

Technical Perspective Broadening and Deepening Query Optimization Yet Still Making Progress

By Jeffrey F. Naughton

P. 81

Multi-Objective Parametric Query Optimization

By Immanuel Trummer and Christoph Koch

P. 90

Technical Perspective Shedding New Light on an Old Language Debate

By Jeffrey S. Foster

P. 91

A Large-Scale Study of Programming Languages and Code Quality in GitHub

By Baishakhi Ray, Daryl Posnett,
Premkumar Devanbu, and Vladimir Filkov



**ACM Conference
Proceedings
Now Available via
Print-on-Demand!**

*Did you know that you can
now order many popular
ACM conference proceedings
via print-on-demand?*

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

**For available titles and
ordering info, visit:
librarians.acm.org/pod**



DOI:10.1145/3068610

To view the accompanying paper, visit doi.acm.org/10.1145/3068612

Technical Perspective

Broadening and Deepening Query Optimization Yet Still Making Progress

By Jeffrey F. Naughton

QUERY OPTIMIZATION IS a fundamental problem in data management. Simply put, most database query languages are declarative rather than imperative—that is, they specify properties the answer should satisfy, rather than give an algorithm to compute the answer. The best known and most widely used database query language—SQL—is a prime example of a language for which optimization is essential.

By “essential,” I mean that database optimization is not a matter of shaving 10% or even a factor of 2x from a query’s execution time. In database query evaluation, the difference between a good plan and a bad or even average plan can be multiple orders of magnitude—so successful query optimization makes the difference between a plan that runs quickly and one that never finishes at all. Accordingly, since the seminal papers in the 1970s, query optimization has received and continues to receive a great deal of attention from both the industrial and research database communities.

Early work on optimization focused on a scenario in which the query was fully specified, and the optimization goal was query evaluation time. That is, the problem was: What is the fastest way to evaluate this query? While this problem was (and is!) challenging, it is not broad enough to capture the optimization problem faced by modern systems. As an important example, many times the query is not fully specified in advance (for example, it may contain variables, or “parameters” that are only discovered at runtime). This generalization gives rise to *parametric query optimization*, where the problem is as follows: Given a partially specified query, find a set of good evaluation plans, one of which will be chosen at runtime when the parameter is instantiated.

Yet another necessary generalization involves the optimization goal. Some-

times execution time is not the only criterion by which plans should be selected. As a prominent and current example, if the query is being run in the cloud, the system may obviously want to find fast evaluation plans, but may also desire inexpensive ones. That is, now we have two objectives: running time and cost. This gives rise to *multi-objective query optimization*, where the problem is: Given a query and a set of objectives, find a set of plans that are Pareto-optimal for these objectives (a plan is “Pareto-optimal” if it is not dominated in all objectives by other plans.)

Both parametric and multi-objective query optimization have been studied in the past, but the following paper by Trummer and Koch is a remarkable tour de force exploration of the combination of the two. Here, the problem is the following: Given a partially specified query, and multiple objectives for the resulting plan, find a set of Pareto-optimal plans that can be chosen at runtime by filling in all parameters.

Since the original query optimization problem and its variants are already very difficult, one might despair that simultaneously treating two substantial extensions would yield a hopelessly intractable problem. This paper is surprising in its elegance and effectiveness. It embeds the problem in an insightful and expressive formal framework, and specifies a solution that combines aspects of piecewise linear functions, dynamic programming with pruning based upon Pareto polytope analyses, and linear programming. A thorough set of experiments with an implementation of their algorithm completes the paper, indicating all of this actually works.

Jeffrey F. Naughton (naughton@google.com) is a principal scientist at Google and previously a professor of computer science at the University of Wisconsin, Madison.

Copyright held by author.

Multi-Objective Parametric Query Optimization

By Immanuel Trummer and Christoph Koch

Abstract

We propose a generalization of the classical database query optimization problem: multi-objective parametric query (MPQ) optimization. MPQ compares alternative processing plans according to multiple execution cost metrics. It also models missing pieces of information on which plan costs depend upon as parameters. Both features are crucial to model query processing on modern data processing platforms.

MPQ generalizes previously proposed query optimization variants, such as multi-objective query optimization, parametric query optimization, and traditional query optimization. We show, however, that the MPQ problem has different properties than prior variants and solving it requires novel methods. We present an algorithm that solves the MPQ problem and finds, for a given query, the set of all relevant query plans. This set contains all plans that realize optimal execution cost tradeoffs for any combination of parameter values. Our algorithm is based on dynamic programming and recursively constructs relevant query plans by combining relevant plans for query parts. We assume that all plan execution cost functions are piecewise-linear in the parameters. We use linear programming to compare alternative plans and to identify plans that are not relevant. We present a complexity analysis of our algorithm and experimentally evaluate its performance.

1. INTRODUCTION

1.1. Context

The goal of the database query optimization is to map a query (describing the data to generate) to the optimal query plan (describing how to generate the data). Query optimization is a long standing research area in the database field dating back to the 1970s.¹⁴ The original query optimization problem model has been motivated by the capabilities of data processing systems at that time. However, there have been fundamental advances in data processing techniques and systems in the meantime. Hence the original problem model is not sufficiently expressive to capture all relevant aspects of modern data processing systems. In this paper, we propose an extension of the classical query optimization problem model and a corresponding optimization algorithm.

Alternative query plans are compared according to their execution cost (e.g., execution time) in query optimization. Query optimization variants can be classified according to how they model the execution cost of a single query plan. Traditional query optimization¹⁴ models the cost of a query plan as scalar cost value $c \in \mathbb{R}$. This implies that query plans are compared according to one single cost metric. It also

implies that all information required to produce cost estimates is available to the query optimizer. The goal in classical query optimization is to find a query plan with minimal execution cost.

Multi-objective query optimization^{1, 7, 11, 16, 17} generalizes the classical model and associates each query plan with a cost vector $c \in \mathbb{R}^n$ instead of a scalar value. This allows to model scenarios where multiple execution cost metrics are of interest. If data processing takes place in the cloud then we are not only interested in execution time but also in monetary execution fees. Different components of the plan cost vector represent cost according to different cost metrics. The goal is to find the set of Pareto-optimal query plans for which no alternative plan offers better cost according to all metrics.

Parametric query optimization^{3, 4, 6, 8, 10, 13} generalizes the standard model in a different way. It associates each query plan with a cost function $c \in \mathbb{R}^m \rightarrow \mathbb{R}$, mapping from a multidimensional parameter space to a one-dimensional cost space. Parameters represent pieces of information that are not yet available at optimization time but required to estimate plan execution cost. For instance, parametric query optimization allows to optimize query classes that are defined via query templates with unspecified predicates. One parameter could then represent the selectivity of one unspecified predicate. The concrete predicate becomes known not before run time and so does the concrete parameter value. The goal in parametric query optimization is typically to find a set of plans containing for each possible parameter value combination, the plan with minimal execution cost.

1.2. Problem

We propose multi-objective parametric query (MPQ) optimization, a query optimization variant that generalizes multi-objective query optimization, parametric query optimization, and classical query optimization at the same time. MPQ models the cost of a single query plan as a cost function $c \in \mathbb{R}^m \rightarrow \mathbb{R}^n$ that maps a multidimensional parameter space to a multidimensional cost space. MPQ assumes that query plans are compared according to multiple cost metrics and that cost estimates depend on parameters whose values are unknown at optimization time.

The goal in MPQ is to find the set of Pareto-optimal plans for each possible parameter value combination. This problem model is required wherever the application scenarios of multi-objective query optimization intersect with the ones

The original version of this article was published in the *Proceedings of the VLDB Endowment*, Volume 8.

of parametric query optimization. The following example describes a scenario in which MPQ is necessary.

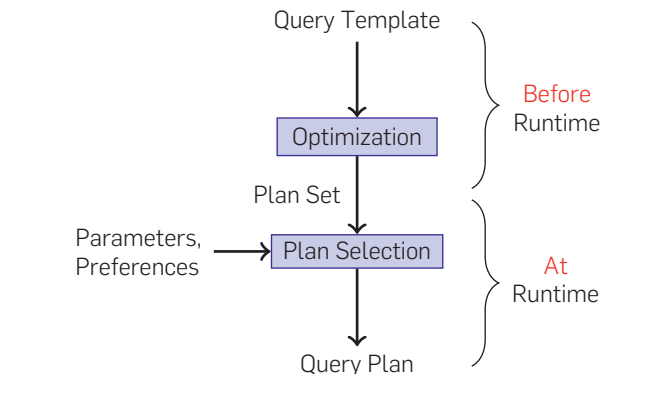
EXAMPLE 1. Assume that we need to process the same query in regular time intervals. Query processing takes place in the cloud and we would like to use Amazon EC2 Spot Instances. Here, we care about two execution cost metrics which is the execution time and the monetary execution fees. We can trade between them by adapting the type and number of the computational resources that we rent from Amazon. The query processing cost also depends on parameters: the pricing of Amazon Spot Instances. Due to unpredictable fluctuations, we do not know their values in advance. As we process the same query repeatedly, we can determine the set of all potentially relevant query plans in a preprocessing step. At run time, given concrete Spot prices and execution cost bounds, we can efficiently select the best query plan out of the precomputed set. This avoids expensive optimization at run time. The preprocessing step requires MPQ since multiple plan cost metrics and parameters need to be considered.

There are many other scenarios in which multiple processing cost metrics are of interest. Techniques for approximate query processing allow us to trade between execution time and result precision.¹ Different query plans can realize different tradeoffs between energy consumption and execution time for the same query.¹⁸ If data is processed by crowd workers then latency, execution fees, and result precision are all relevant cost metrics.¹² If the queries we want to process at run time correspond to query templates that are known before run time then we can make query optimization a preprocessing step. At preprocessing time, plan cost estimates depend on parameters with unknown values. Those parameters can represent query properties which are not fully specified in the template or properties of the query execution platform (e.g., the Spot Instance prices) that will become known only at run time. MPQ is applicable in such scenarios and avoids query optimization at run time.

The result of MPQ is the set of all potentially relevant query plans for a given query or query template. It contains all Pareto-optimal plans for each possible parameter value combination. At run time, we can select the best query plan out of that set based on the concrete parameter values and based on user preferences. Users can specify their preferences in advance (e.g., by specifying cost bounds and priorities between different cost metrics^{14,16}) such that the optimal plan according to those preferences can be selected automatically. As an alternative, we can use the precomputed plan set to visualize all Pareto-optimal cost tradeoffs for given parameter values. This allows users to select the preferred cost tradeoff directly.¹⁷ Figure 1 illustrates the context of MPQ.

So far we have introduced a very generic problem model for MPQ. In order to make the problem tractable, we restrict ourselves to a specific class of cost functions in this paper: we consider piecewise-linear plan cost functions. Many approaches for parametric query optimization^{6,8} consider only piecewise-linear plan cost functions as well since such

Figure 1. Multi-objective parametric query optimization precomputes a set of relevant query plans. The optimal plan is selected from that set according to parameter values and user preferences.



functions can approximate arbitrary functions.⁸ The difference between our model and the one used in parametric query optimization is that we associate each plan with multiple piecewise-linear cost functions representing cost according to different metrics.

1.3. Algorithm

We present an algorithm that solves MPQ for piecewise-linear plan cost functions. Our algorithm is based on dynamic programming. It recursively decomposes the input query for which we need to determine the set of relevant query plans into subqueries. In a bottom-up approach, it recursively calculates sets of relevant plans for a query out of optimal plan sets for its subqueries: it combines plans that are relevant for the subqueries to form new plans that are potentially relevant for the decomposed query. Dynamic programming is a classical approach for query optimization. The crucial difference between our algorithm and prior algorithms is the implementation of the pruning function, that is, in how we compare alternative query plans and prune out suboptimal plans.

Conceptually, we associate each plan for a query or subquery with a region in the parameter space for which the plan is Pareto-optimal. We call this region the Pareto region. The goal during pruning is to compare alternative plans generating the same result in order to discard suboptimal plans. We compare plans pair-wise and determine for each plan the parameter space region in which it is dominated by another plan, that is, in which the other plan has comparable or better cost according to each plan cost metric. Then we reduce the Pareto region of the first plan by the region in which it is dominated. If the Pareto region of a plan becomes empty then it is not Pareto-optimal for any parameter value combination and can be discarded.

All Pareto regions that could ever occur during the execution of that algorithm can be represented using the following formalism. We represent Pareto regions as a union of convex polytopes in the parameter space from which other convex polytopes have been subtracted. We prove that this representation is closed under all operations that the algorithm needs to perform on Pareto regions. Note that this

region shape is a consequence of the class of cost functions (piecewise-linear functions) that we consider.

The algorithm needs to perform several elementary operations on Pareto regions and cost functions. For instance, it must verify whether a Pareto region is empty or calculate a parameter space region in which one plan is preferable to a second one. We show how all those operations can be implemented based on the aforementioned representation of Pareto regions. We implement those operations using linear programming.

1.4. Outline

The remainder of this paper is organized as follows. We define the MPQ problem and related concepts more formally in Section 2. In Section 3, we describe our algorithm for MPQ with piecewise-linear plan cost functions. In Section 4, we analyze the MPQ problem and the asymptotic complexity of our algorithm. We present experimental results for an implementation of our algorithm in Section 5 and discuss related work in Section 6.

2. FORMAL PROBLEM STATEMENT

We now define the MPQ problem. A query describes data to generate. The description of our algorithm for solving MPQ problems, given in the next section, focuses on simple SQL join queries. An SQL join query is defined by a set of tables to join. A subquery joins a subset of tables. Standard methods exist by which a query optimization algorithm for this simple query language can be extended into an algorithm supporting full SQL queries.¹⁴

A query plan describes how to generate data. We say that a query plan answers a query if it generates the data, that is, described by the query. We assume in the following that query plans consist of a sequence of scan operations and binary join operations. For a query q , we denote by $P(q)$ the set of alternative plans that answer the query.

We compare query plans according to their execution cost. The cost of a given plan depends on a set of real-valued parameters. The set of parameters is a property of the query. All alternative plans in $P(q)$ depend therefore on the same parameters. A parameter value vector contains for each parameter a corresponding value. We do not know the parameter values at optimization time. The parameter space is the set of all possible parameter value vectors. We assume in the following that there are n parameters and denote by $X \subseteq \mathbb{R}^n$ the n -dimensional parameter space. A parameter space region is a subset of the parameter space.

We compare query plans according to multiple execution cost metrics. A cost vector contains for each cost metric a nonnegative cost value. We assume in the following that there are m execution cost metrics and denote by $C = \mathbb{R}^m$ the space of cost vectors. We associate each query plan p with a cost function $c_p : X \rightarrow C$ that maps the n -dimensional parameter space to the m -dimensional cost space. We can compare the cost of query plans for specific parameter values. Denote by $x \in X$ a parameter value vector and by p_1 and p_2 two plans answering the same query. We say that p_1 dominates p_2 for x , written $p_1 \preceq_x p_2$, if p_1 has lower or equivalent cost than p_2 according to each metric for parameter values x .

In other words, p_1 dominates p_2 if $c_{p_1}(x)$ contains for no component a higher value than $c_{p_2}(x)$. Now, we are ready to introduce the MPQ problem.

DEFINITION 1. An *MPQ problem* is defined by a query q , a parameter space X , and a cost space C . A solution is a subset $S \subseteq P(q)$ of query plans such that for each possible plan $p \in P(q)$ and for each possible parameter value vector $x \in X$ there is a solution plan $s \in S$ such that s dominates p for x , that is, $s \preceq_x p$.

We focus on a subclass of MPQ problems that restricts the class of cost functions. In order to define the class of cost functions that we consider, we must first introduce convex polytopes. A convex polytope is defined by a set of linear inequalities. The convex polytope is the set of points in the parameter space that satisfy all its inequalities. We use the terms convex polytope and polytope as synonyms. A linear cost function is defined by a constant b and an n -dimensional weight vector $w \in \mathbb{R}^n$ such that $b + w^T \cdot x$ is the associated cost value for each parameter vector $x \in X$. A scalar piecewise-linear cost function is a cost function that allows to partition the parameter space into convex polytopes such that the function is linear in each polytope. A vector-valued piecewise-linear cost function consists of one piecewise-linear cost function for each cost metric. We use the terms vector-valued piecewise-linear cost function and piecewise-linear cost function as synonyms. We restrict our scope to MPQ with piecewise-linear cost functions.

3. ALGORITHM

Our algorithm produces a set of relevant plans for a given query. A plan is relevant if its execution cost is Pareto-optimal for some parameter value combination.

3.1. Overview

Our algorithm splits the input query recursively into smaller and smaller parts until we obtain atomic subqueries. We start with atomic subqueries and calculate the set of relevant plans for each of them. After that, larger subqueries are treated. We treat subqueries in an order which makes sure that before treating a query, we have calculated relevant plan sets for each of its subqueries. The reason for restricting the order is that we want to calculate the set of relevant plans for a query out of the sets of relevant plans for its subqueries. More precisely, we can guarantee that each relevant plan for a query can be obtained by splitting the query into two subqueries and combining a relevant plan for the first subquery with a relevant plan for the second subquery, thereby generating a new query plan. Having calculated the set of relevant plans for each subquery, we can therefore obtain a superset of relevant query plans by iterating over all possible splits into subqueries and over all possible combinations of relevant subplans. In order to reduce the superset to the actual set of relevant query plans, we must prune plans answering the same query. Pruning them means to identify and to discard plans that are irrelevant. The input query is treated last. The set of relevant plans for the input query is

the desired algorithm output. In summary, our algorithm can be written as follows:

- Iterate over all subqueries s of the input query in ascending order of query size:
 - If subquery s is an atomic subquery then consider all possible plans for s .
 - Otherwise, if s is not an atomic subquery, then iterate over all possibilities to decompose s into two subqueries s_1 and s_2 :
 - For each split into two subqueries s_1 and s_2 , consider all plans that are combinations of a relevant plan for s_1 and a relevant plan for s_2 .
 - Prune all considered plans to obtain the set of relevant plans for s .

As many query optimization algorithms,^{8, 14, 16} our algorithm is based on dynamic programming. We can use dynamic programming since the principle of optimality holds for query optimization.¹⁴ Formulated in general terms, the principle of optimality designates the problem property that optimal solutions can be obtained by combining optimal solutions to subproblems. In the context of query optimization, the principle of optimality means more specifically that optimal query plans can be obtained by combining optimal plans for subqueries. The principle of optimality has been shown to hold for all common execution cost metrics in multi-objective query optimization.¹⁶ This means that a Pareto-optimal query plan can be combined from Pareto-optimal plans for subqueries. A relevant plan is Pareto-optimal for some points in the parameter space. It is therefore intuitive that a relevant query plan can be combined from relevant plans for the subqueries (we omit the formal proof). In other words, the principle of optimality holds for MPQ as well. It is the fundament of our MPQ algorithm.

3.2. Pruning

Many query optimization algorithms for classical query optimization,¹⁴ multi-objective query optimization,¹⁶ or parametric query optimization⁸ are based on dynamic programming. The primary difference between all those algorithms is the realization of the pruning function. As we treat a novel problem variant, we must design a novel pruning function. In the following, we describe how our algorithm prunes query plans, that is, how it compares plans for the same query and identifies irrelevant plans.

Our pruning function is based on the key concept of the Pareto region. Each query plan is associated with a Pareto region. This is a parameter space region in which it realizes Pareto-optimal cost tradeoffs. A plan is irrelevant if its Pareto region is empty. The goal of the pruning function is to compare a set of plans answering the same query in order to calculate their Pareto regions. The pruning function works as follows. At pruning start, we assume by default that each plan is Pareto-optimal in the entire parameter space. This means that we assign the entire parameter space as Pareto region to each query plan. During pruning, we compare all query plans answering the same query pair-wise in order to calculate their true Pareto regions. If we compare two

plans p_1 and p_2 and we find that plan p_1 has better cost than or equivalent cost to p_2 according to all cost metrics for the parameter space region X then we reduce the Pareto region of p_2 by subtracting X . Pareto regions can only shrink during a pruning operation. Once the region of one plan becomes empty, it is irrelevant and can be safely discarded. We discard plans as soon as possible in order to avoid unnecessary comparisons.

More precisely, the pruning function iterates over all plan pairs and executes for each pair the following steps. First, it identifies the region in which one plan dominates the other plan. Second, it updates the Pareto region of the dominated plan by subtracting the region in which it is dominated. Third, it checks whether the Pareto region of the dominated plan becomes empty after the update. In that case, the plan is discarded and does not participate in further comparisons. Figure 2 illustrates how the Pareto region of a plan is reduced after comparing it to another plan. The example refers to a scenario where two parameters and two cost metrics are considered (execution time and fees).

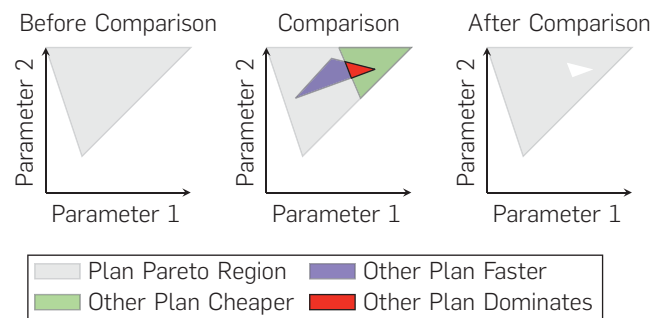
Note that two plans can mutually dominate each other in different parameter space regions. Having determined that a first plan dominates a second plan for some parameter space region, we must therefore still verify if the second plan dominates the first plan as well.

3.3. Data structures

We describe the data structures by which we represent plan cost functions and Pareto regions. Our plan cost model is based on piecewise-linear functions. A piecewise-linear function is linear in parameter space regions that form convex polytopes. A linear function can be represented by a constant and by weights capturing the slope of the function for each parameter. Hence a piecewise-linear function can be represented by a set of convex polytopes where each convex polytope is associated with a constant and weights. We consider multiple plan cost metrics. Each query plan is therefore associated with one piecewise-linear cost function per plan cost metric.

We consider the class of piecewise-linear cost functions to represent plan cost. We decided to use that class of functions since it allows to approximate arbitrary functions

Figure 2. We subtract the area in which a plan is dominated from its Pareto region.



up to an arbitrary degree of precision (using more pieces increases precision). In contrast to that, we cannot freely decide which class of shapes we consider for representing Pareto regions. The algorithm must be able to represent each shape that could potentially occur during pruning. Our decision to use piecewise-linear cost functions implies the class of shapes that we need to consider as Pareto regions.

We describe our representation of Pareto regions. We motivate this representation in an informal way. It is, however, relatively easy to prove that the proposed representation covers all possible cases.

We start by considering the special case of linear cost functions. Parametric query optimization is a special case of MPQ. It has been shown in the domain of parametric query optimization that the parameter space region in which one plan is better than another plan according to one cost metric is a convex polytope if both plans have linear cost functions.⁶ In a setting with multiple cost metrics, a plan is strictly better than another plan if it is better according to each cost metric. The region in which a plan is better than another one is therefore an intersection of multiple convex polytopes. An intersection of convex polytopes is a convex polytope again. The region in which all other plans are better than a given plan is, hence, a union of convex polytopes.

Now, let us generalize that reasoning from linear cost functions to piecewise-linear cost functions. The generalization is straightforward. Given two piecewise-linear cost functions, we can always partition the parameter space into convex polytopes such that both cost functions are linear in each polytope. Thereby we reduce the case of piecewise-linear cost functions to the case of linear cost functions. In summary, we can represent the Pareto region of a plan as a union of convex polytopes from which we subtract another union of convex polytopes.

3.4. Elementary operations

Having described the data structures used to represent cost functions and Pareto regions, we outline now how to implement elementary operations on those data structures. We require the following elementary operations to realize the pruning function as described before. First, given the cost functions of two plans, we must determine the parameter space region in which one plan dominates the other one. Second, given a Pareto region of a plan and a region in which it is dominated, we must reduce the Pareto region by that region. Third, given a Pareto region, we must determine whether it is empty.

Convex polytopes are described by a set of linear inequalities and we consider linear cost functions. All elementary operations that we describe in the following can, hence, be realized by solving systems of linear inequalities. Executing the elementary operations therefore requires a linear solver.

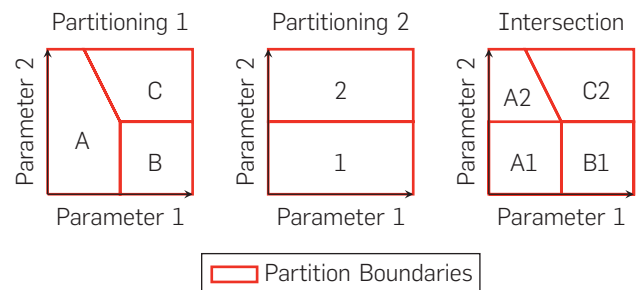
We describe how to determine the parameter space region in which one plan dominates another one. Assume first that we have only one cost metric and that cost functions are linear. Then, we can directly use the linear solver

to determine the parameter space region in which one function has lower values than the other one. Now, we generalize from linear cost functions to piecewise-linear functions. Each piecewise-linear function partitions the parameter space into convex polytopes in which the function is linear. If we compare two piecewise-linear functions then we can partition the parameter space such that both functions are linear in each partition. More precisely, we obtain the aforementioned partitioning by intersecting the partitions associated with the first cost function with the partitions associated with the second function. Figure 3 illustrates how we intersect two parameter space partitionings in a two-dimensional parameter space. Having this partitioning, we apply the method for linear cost functions separately in each partition. If we have the subregion in which a first plan dominates a second one for each parameter space region then the union of those subregions is the total area in which the first plan dominates. If we have multiple cost metrics instead of only one, then we can apply the method described before for each cost metric separately. If we have for each cost metric the parameter space region in which the first plan dominates the second one then the intersection of those areas (over all cost metrics) yields the area in which the first plan is better according to all cost metrics.

Given the area in which a plan is dominated, we must subtract it from that plan's Pareto region. The implementation of this operation is straightforward: as discussed before, we represent Pareto regions as a union of convex polytopes from which other convex polytopes have been subtracted. The region in which one plan dominates another one must consist of convex polytopes. In order to subtract such a region from the Pareto region, we simply add the corresponding polytopes to the list of subtracted polytopes.

We must determine whether a given Pareto region is empty. A Pareto region is a set of polytopes from which other polytopes have been subtracted. We consider first the special case of one polytope P^+ from which another set of polytopes $\{P_i^-\}$ have been subtracted. We want to verify whether the given polytope becomes empty after the subtractions. We can verify that as follows. Assume that all subtracted polytopes P_i^- are contained within P^+ . Then the region remaining after subtraction becomes empty if and only if $\cup_i P_i^- = P^+$. We can use the algorithm by Bemporad et al.² to check the latter

Figure 3. To compare two piecewise-linear cost functions, we intersect the parameter space partitions in which each function is linear. We compare the functions separately in each of the resulting partitions.



condition. The algorithm by Bemporad verifies whether the union of a given set of convex polytopes forms a convex polytope again. If this is the case then the algorithm constructs that polytope. The condition $\cup_i P_i^- = P^+$ can only be verified if $\cup_i P_i^-$ forms a convex polytope. In that case, the algorithm by Bemporad constructs the polytope $P^- = \cup_i P_i^-$ and a linear solver can verify whether P^- and P^+ are equivalent.

4. ANALYSIS

We analyze the formal properties of the freshly introduced MPQ problem in this section. We also analyze the complexity of the algorithm described in the last section.

4.1. Problem analysis

MPQ generalizes parametric query optimization since it allows to consider multiple plan cost metrics instead of only one. We compare the formal properties of MPQ to the properties of parametric query optimization in the following.

The parametric query optimization problem with linear cost functions has the following property: if the same query plan is optimal at all vertices of a convex polytope in the parameter space then that plan must be optimal inside the polytope as well.⁶ This property is commonly known as one of the “guiding principle of parametric query optimization.”⁵ Many algorithms for parametric query optimization exploit this property as follows^{6,9}: they recursively decompose the parameter space into convex polytopes and calculate optimal query plans at the vertices. Due to the guiding principle, the decomposition of the parameter space can be stopped once the same plan is optimal at all vertices of a polytope. Such algorithms transform the parametric query optimization problem into a series of traditional query optimization problems (calculating the optimal plan at a polytope vertex is a traditional query optimization problem). This has the advantage that traditional query optimizers can be used for parametric query optimization with minimal changes. It is therefore interesting to verify whether an analog property holds for MPQ.

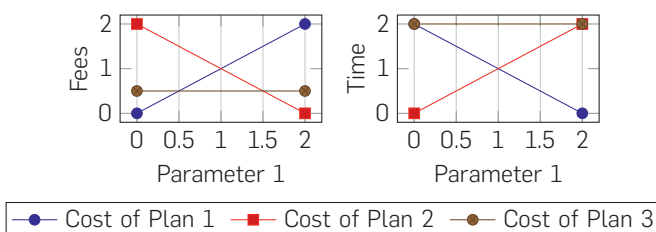
Unfortunately this is not the case as we show next. The following property for MPQ would be analog to the guiding principle of parametric query optimization: if the same set of plans is Pareto-optimal at all vertices of a polytope in the parameterspace then that set of plans must be Pareto-optimal inside the polytope as well. Figure 4 illustrates a counter example showing that this property does not hold. The figure

refers to a scenario in which two cost metrics, namely execution time and execution fees, are of interest. Cost functions depend on a single parameter, called “Parameter 1” in the figure, that could refer to unspecified predicates in the input query template. We see the cost functions of three plans. For parameter value 0, plan 1 is Pareto-optimal since it has lowest execution fees. Plan 3 is Pareto-optimal since it has lower execution time than all other plans. Plan 2 is, however, dominated by plan 1 since plan 1 has equivalent execution time and lower execution cost. This means that plan 2 is not Pareto-optimal for parameter value 0. For parameter value 2, the situation is similar and plans 1 and 3 are Pareto-optimal while plan 2 is not. For parameter values between 0.5 and 1.5, plan 2 is however Pareto-optimal. Even though the same set of plans is Pareto-optimal at the borders of the parameter value interval $[0, 2]$, additional plans can be Pareto-optimal for values at the interior of that interval. All plan cost functions are linear in the example and an interval is a special case of a convex polytope. The example is minimal for MPQ: having less than two cost metrics would lead to parametric query optimization. Having less than one parameter would lead to multi-objective query optimization. Hence, we can conclude from this example that the guiding principles do not apply for MPQ in general.

4.2. Algorithm analysis

The space and time complexity of dynamic programming-based query optimization algorithms depends on the number of plans stored per subquery. In traditional query optimization, plans are compared according to one cost metric and cost functions do not depend on parameters. If we assume that alternative query plans are compared based on their cost values alone then exactly one plan, a plan with minimal cost, remains after pruning an arbitrary set of plans. In parametric query optimization, plans are compared according to one cost metric but cost functions depend on parameters. This means that different plans can be optimal for different parameter values. In multi-objective query optimization, we compare plans according to different cost metrics. Hence multiple plans can be Pareto-optimal for each subquery. As a result, we generally need to store multiple plans per subquery in parametric and in multi-objective query optimization. The number of plans to store depends on many factors. Research in parametric query optimization has focused on analyzing how the number of plans per subquery depends on the number of parameters. Research in multi-objective query optimization has focused on the dependency between the number of plans and the number of cost metrics. Such analysis is necessarily based on simplifying assumptions. Traditionally, the weights that define the cost functions of different query plans are assumed to follow independent random distributions.^{6,7} Based on that assumption, the number of remaining plans after pruning can be considered a random variable as well and we can calculate its expected value. This reasoning led to an asymptotic upper bound of 2^m , where m designates the number of cost metrics, on the expected number of plans per subquery in multi-objective query optimization.⁷

Figure 4. The guiding principle of parametric query optimization does not hold for multi-objective parametric query optimization.



We perform a similar analysis to determine the expected number of plans per subquery in MPQ. We consider linear cost functions. We denote the number of parameters by n . A linear function is therefore defined by a vector consisting of $n + 1$ components, specifying the function slope for each parameter and a constant. We still denote the number of considered plan cost metrics by m . Each query plan is therefore associated with m linear functions. The multi-dimensional cost function of each query plan can therefore be described by a matrix containing $m \cdot (n+1)$ components, specifying for each cost metric the cost slopes and a constant. Assume that we have two cost functions and that all constants and slopes describing the first function are lower than the corresponding entries for the second cost function. Then the first cost function has for each cost metric a lower constant cost component and a lower slope in each parameter. In other words, the first cost function has lower values than the second one for arbitrary parameter values and cost metrics. If both cost functions are associated with query plans then the plan associated with the second function is clearly irrelevant.

We can exploit this fact as follows. Assume that we choose an arbitrary number of D -dimensional vectors randomly with independent identical distribution. Then the expected number of vectors such that no other vector has a lower or equivalent value in each component is bounded by 2^D .⁷ We assume that vectors describing the cost functions of different query plans are chosen randomly with independent and identical distribution. Setting $D = m \cdot (n + 1)$, we infer that the expected number of vectors such that no other vector has lower or equal values in all components is bounded by $2^{m \cdot (n+1)}$. As outlined before, this is at the same time an upper bound on the expected number of relevant query plans per subquery.

In order to obtain an upper bound on the asymptotic space complexity, we multiply the aforementioned bound by the number of subqueries. We generate new plans by combining two relevant plans. The number of generated plans grows therefore as the square of the number of relevant plans. All generated plans for the same subquery are compared pair-wise during pruning. The number of plan comparisons grows therefore as the fourth power of the number of relevant plans. Multiplying by the number of subquery splits yields the time complexity measured by the number of plan comparisons.

5. EXPERIMENTS

5.1. Experimental setup

We evaluate our MPQ algorithm experimentally. More precisely, we study how optimization time depends on the input query size and on the number of considered parameters. Our experiments are based on an example scenario in which SQL queries are processed in the cloud. Hence, we compare alternative query plans according to two cost metrics: execution time and monetary execution fees. We consider a restricted class of SQL queries: each query is described by a set of tables to join, by predicates defined on single tables, and by binary join predicates defined on table pairs. We assume that our MPQ algorithm is applied to

query templates which are not fully specified: the predicates defined on single tables are placeholders. The selectivity of such a predicate, meaning the average fraction of tuples satisfying the predicate, is unknown to our MPQ algorithm. Hence, the selectivity of each predicate placeholder must be represented by a parameter. Our algorithm finds all plans realizing optimal cost trade-offs for each possible parameter value combination.

We generate the queries for our benchmark randomly. We use the method described by Steinbrunn et al.¹⁵ to produce random queries that join a given number of tables. The number of rows in each table and the selectivity of each predicate is chosen randomly according to that method. We distinguish two classes of queries: chain queries and star queries. For chain queries, the binary join predicates connect query tables in a chain. For star queries, the binary join predicates connect one table (the middle of the “star”) to all other query tables. The number of predicates is for both query classes one less than the number of tables.

We describe the plan search space that our algorithm considers. Our algorithm considers all possible orders in which tables can be joined with only one restriction: whenever we have the choice between joining two relations that are connected via a binary join predicate and joining two relations where this is not the case then only joins of the first category are considered. This restriction on the join order is often used in query optimization.^{14, 15} In addition to the join orders, our algorithm considers different scan and join operators. For scanning single tables on which a predicate is defined, we consider a full scan and an index-based scan. Which of the two operators is preferable depends on the selectivity of the predicate. If the selectivity is low (few tuples will satisfy the predicate) then the index scan is often preferable. If the predicate is satisfied for most tuples then the full scan is more efficient. We model the selectivity of a predicate defined on a single table by a parameter. The optimal choice for the scan operator therefore depends on the value of that parameter. We consider two join operators: a distributed join and a single-node hash join. For sufficiently large amounts of input data, the distributed join saves execution time. On the other side, the distributed join requires to rent more computational resources from the cloud provider and is therefore more expensive. Hence, we can realize different tradeoffs between execution time and execution fees by selecting between alternative join operators. We implemented our MPQ algorithm in Java 1.7. We used Gurobi 5.6^a as linear solver. All experiments were executed on an iMac equipped with an i5-3470S processor with 2.9 GhZ and 16 GB of RAM.

5.2. Experimental results

Figure 5 shows our experimental results. Each data point in that figure corresponds to the median value of 25 randomly generated test cases. We report optimization time, the number of generated query plans (counting plans for the input

^a <http://www.gurobi.com/>.

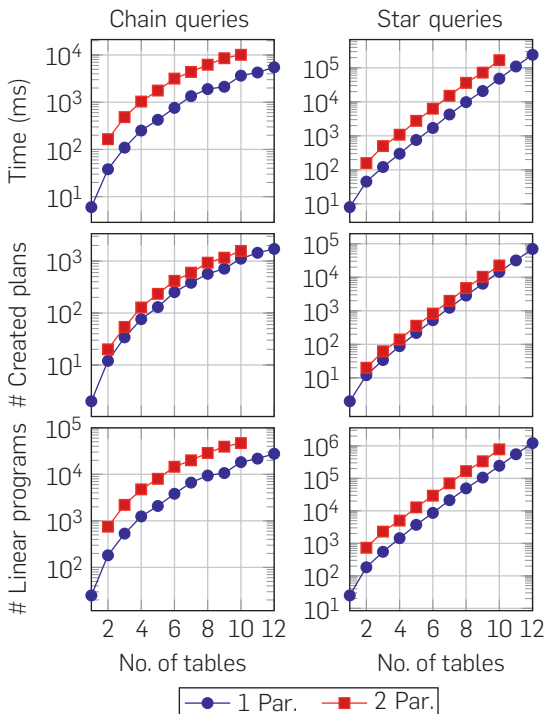
query and plans for subqueries), and the number of solved linear programs. We generated query templates joining between 2 and 12 tables and having between one and two parameters.

Optimization time increases in the number of tables. As predicted by our formal analysis in the previous section, optimization time also increases in the number of parameters. Optimization time grows faster in the number of query tables for star queries than for chain queries. The reason is that the number of admissible join orders grows faster in the number of query tables for star queries. Speaking of admissible join orders, we mean join orders that comply with the restriction mentioned before. Optimization time, the number of generated plans, and the number of solved linear programs are all correlated. This is intuitive as the number of generated plans relates to the number of plan comparisons that are required during pruning. The number of linear programs is related to the number of plan comparisons since plan comparisons are realized by solving linear programs. The time required for generating plans and for solving linear programs adds to optimization time.

The query sizes that we consider in our benchmark are typical for query sizes as they appear in standard benchmarks: the queries in the popular TPC-H benchmark^b join at most eight tables. MPQ takes longer than traditional query optimization. In contrast to traditional query optimization, MPQ takes, however, place before runtime. This makes higher optimization times acceptable.

^b <http://www.tpc.org/tpch/>.

Figure 5. Optimization time, number of generated plans, and number of solved linear programs.



6. RELATED WORK

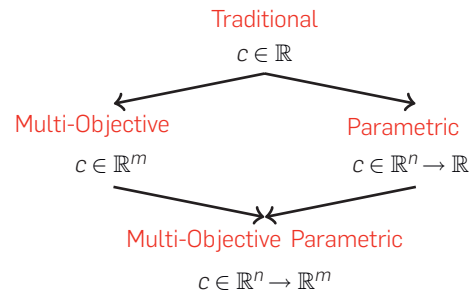
Figure 6 shows how MPQ optimization relates to prior query optimization variants. The figure shows for each variant the type of cost function c , that is, associated with each query plan. Arrows point from a more restricted to a more general query optimization variant. Multi-objective query optimization^{1, 7, 11, 16, 17} and parametric query optimization^{3, 4, 6, 8, 10, 13} both generalize the traditional query optimization model.¹⁴ Multi-objective parametric query optimization generalizes both of the aforementioned variants.

The algorithm that we propose in this paper allows to solve query optimization problems that prior algorithms cannot solve. Algorithms for parametric query optimization are not applicable to MPQ since they cannot handle multiple cost metrics. Algorithms for multi-objective query optimization are not applicable to MPQ since they cannot handle parameters. Note that parameters and cost metrics have a different semantic such that it is not possible to model parameters as cost metrics or vice versa. Intuitively, we want to “cover” the entire parameter space (by finding plans for each possible parameter value combination) while we do not want to cover the entire cost space (plans with higher cost values than necessary are not part of the result plan set).

The algorithm that we describe in this article is based on dynamic programming. It calculates optimal plans for a query by combining optimal plans for its subqueries. Many query optimization algorithms for traditional query optimization,¹⁴ multi-objective query optimization,^{16, 17} and parametric query optimization⁸ use the same dynamic programming scheme. The difference between our algorithm and all prior algorithms lies in the implementation of the pruning function. We use linear programming in the pruning function. Our algorithm shares this property with prior algorithms for parametric query optimization.⁸ We support however multiple cost metrics and hence the definition of the pruning function, the type of the used data structures, and the implementation of elementary operations on those data structures differ.

Many algorithms for parametric query optimization are based on the guiding principles of parametric query optimization.⁵ They partition the parameter space in a more and more fine-grained manner until a single query plan is optimal in each partition.^{6, 9} The condition that allows to verify whether a single query plan is optimal in a given partition is

Figure 6. Multi-objective parametric query optimization generalizes the cost models of multi-objective and of parametric query optimization.



based on the guiding principles. We have shown in Section 4 that the multi-objective analog of the guiding principles does not hold for MPQ. Hence, we cannot use generalizations of the aforementioned decomposition methods for MPQ.

7. CONCLUSION

The traditional query optimization model is outdated. We proposed a generalized problem model that allows to represent multiple plan cost metrics and multiple parameters. We described and analyzed a first algorithm that solves the novel optimization problem. 

References

- Agarwal, S., Iyer, A., Panda, A. Blink and it's done: Interactive queries on very large data. In *VLDB* 5, 12 (2012), 1902–1905.
- Bemporad, A., Fukuda, K., Torrisi, F. Convexity recognition of the union of polyhedra. *Comput. Geom.* 18, 3 (2001), 141–154.
- Bizarro, P., Bruno, N., DeWitt, D. Progressive parametric query optimization. *KDE* 21, 4 (2009), 582–594.
- Darera, P., Haritsa, J. On the production of anorexic plan diagrams. *PVLDB* (2007), 1081–1092.
- Dey, A., Bhaumik, S., Haritsa, J. Efficiently approximating query optimizer plan diagrams. In *VLDB* 1, 2 (2008), 1325–1336.
- Ganguly, S. Design and analysis of parametric query optimization algorithms. In *VLDB* (1998), 228–238.
- Ganguly, S., Hasan, W., Krishnamurthy, R. Query optimization for parallel execution. In *SIGMOD* (1992), 9–18.
- Hulgeri, A., Sudarshan, S. Parametric query optimization for linear and piecewise linear cost functions. In *VLDB* (2002), 167–178.
- Hulgeri, A., Sudarshan, S. AniPQO: Almost non-intrusive parametric query optimization for nonlinear cost functions. In *VLDB* (2003), 766–777.
- Ioannidis, Y.E., Ng, R.T., Shim, K., Sellis, T.K. Parametric query optimization. *VLDBJ* 6, 2 (May 1997), 132–151.
- Papadimitriou, C., Yannakakis, M. Multiobjective query optimization. In *PODS* (2001), 52–59.
- Park, H., Widom, J. Query optimization over crowdsourced data. In *VLDB* (2013), 781–792.
- Reddy, N., Haritsa, J. Analyzing plan diagrams of database query optimizers. *VLDB* (2005), 1228–1239.
- Selinger, P.G., Astrahan, M.M., Chamberlin, D.D., Lorie, R.A., Price, T.G. Access path selection in a relational database management system. In *SIGMOD* (1979), 23–34.
- Steinbrunn, M., Moerkotte, G., Kemper, A. Heuristic and randomized optimization for the join ordering problem. *VLDBJ* 6, 3 (1997), 191–208.
- Trummer, I., Koch, C. Approximation schemes for many-objective query optimization. In *SIGMOD* (2014), 1299–1310.
- Trummer, I., Koch, C. An incremental anytime algorithm for multi-objective query optimization. In *SIGMOD* (2015), 1941–1953.
- Xu, Z., Tu, Y.C., Wang, X. PET: Reducing database energy cost via query optimization. *VLDB* 5, 12 (2012), 1954–1957.

Immanuel Trummer (itrummer@cornell.edu), Cornell University, Computer Science Department, Ithaca, NY.

Christoph Koch (christoph.koch@epfl.ch), Ecole Polytechnique Fédérale de Lausanne, EPFL DATA Lab, Lausanne, Switzerland.

© 2017 ACM 0001-0782/17/10 \$15.00



The FIRST authoritative resource.

EDITED BY

Sharon Oviatt, *Incaa Designs*

Björn Schuller, *University of Passau, Imperial College London*

Philip Cohen, *VoiceBox Technologies*

Daniel Sonntag, *German Research Center for Artificial Intelligence*

Gerasimos Potamianos, *University of Thessaly*

Antonio Krüger, *German Research Center for Artificial Intelligence*



ISBN: 978-1-970001-64-8 DOI: 10.1145/3015783

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/acm>

Technical Perspective

Shedding New Light on an Old Language Debate

By Jeffrey S. Foster

AS COMPUTER SCIENTISTS, we use programming languages to turn our ideas into reality. It is no surprise, then, that programming language design has been a major concern since at least the 1950s, when John Backus introduced FORTRAN, usually considered the first high-level programming language. The revolutionary innovation of FORTRAN—the thing that made it high-level—was that it included concepts, such as loops and complex expressions, that made the programmer's job easier. To put it another way, FORTRAN showed that a programming language could introduce new *abstractions* that were encoded via a compiler, rather than directly implemented in the hardware.

Not long after the introduction of FORTRAN, other programming languages appeared with somewhat different sets of abstractions: John McCarthy's LISP, which introduced functional programming, and Grace Murray Hopper's COBOL, which aimed to support business, rather than scientific or mathematical, applications. Thus, for at least the last 60 years, programmers have been faced with the question: What programming language should I use?

Today, answering this question has only gotten more difficult. Myriad languages have been developed in the last six decades, with at least a few dozen in common usage today. Moreover, even if in a theoretical sense any general-purpose language can implement any algorithm, in practice the different abstractions provided by different languages seem to have a strong influence on programming tasks. The Internet is filled with heated debates about the merits of functional versus imperative programming, the costs versus the benefits of object orientation, and the trade-offs between dynamic and static typing, among many others.

The following paper aims to bring empiricism to this debate by studying whether programming language

choice and code quality are related. To do so, the authors perform an observational study on a corpus of 728 popular GitHub projects, totaling 63 million lines of code. The largest single project considered is Linux, comprising 15+ million lines of code. The authors apply a variety of techniques to extract data from the subject programs to try to answer the question at hand. They rely on GitHub Linguist to identify the primary language of a project. As a proxy for code quality, they count the total number of defect-fixing commits per project, determined by textually searching for one of a fixed set of keywords in the commit messages. And they use supervised machine learning to approximately classify bug reports into categories.

The authors then apply a range of statistical methods and reach four main conclusions. First, they report that projects in Clojure, Haskell, Ruby, and Scala are slightly less likely to have defect-fixing commits, and those in C, C++, Objective-C, and Python are slightly more likely. Second, they report that languages that are functional, disallow implicit type conversion, have static typing, and/or use-managed memory have slightly fewer defects than languages without these characteristics. Third, they report that defect-proneness does not depend on

domain, for example, user application versus library versus middleware. Last, they report that some defect types, such as memory errors, are strongly associated with certain languages.

These are interesting results. While they suggest that the language does indeed matter, almost all of the observed effects are small ... except that some particular language features, such as a lack of memory safety, do have profound effects.

Like any empirical study, the results here have threats to validity: noise in the data, such as the classification of a commit as defect-fixing, is difficult to account for; defects may have been made and fixed without an intervening commit, for example, defects prevented by a static type checker are likely not included; projects vary significantly in software engineering practices, for example, Linux is an outlier, with an extremely large user base with many developers and testers; tool support for different languages varies significantly; there may be a strong relationship between programmer skill and language choice; language design can obviate classes of errors, for example, buffer overflows can occur in C and C++ but not Java; and in practice the choice of programming language is often constrained both by external factors (for example, the language of existing codebases) and the problem domain (for example, device drivers are likely to be written in C or C++). Finally, while the use of regression analysis in such observational studies can control for confounds and strongly suggest relationships, it cannot definitively establish causation. Even so, this paper raises intriguing questions in an effort to shed light on one of the oldest debates in computer science. 

For at least the last 60 years, programmers have been faced with the question: What programming language should I use?

Jeffrey S. Foster (jfoster at cs.umd.edu) is a professor in the computer science department at the University of Maryland, College Park.

Copyright held by author.

A Large-Scale Study of Programming Languages and Code Quality in GitHub

By Baishakhi Ray, Daryl Posnett, Premkumar Devanbu, and Vladimir Filkov

Abstract

What is the effect of programming languages on software quality? This question has been a topic of much debate for a very long time. In this study, we gather a very large data set from GitHub (728 projects, 63 million SLOC, 29,000 authors, 1.5 million commits, in 17 languages) in an attempt to shed some empirical light on this question. This reasonably large sample size allows us to use a mixed-methods approach, combining multiple regression modeling with visualization and text analytics, to study the effect of language features such as static versus dynamic typing and allowing versus disallowing type confusion on software quality. By triangulating findings from different methods, and controlling for confounding effects such as team size, project size, and project history, we report that language design does have a significant, but modest effect on software quality. Most notably, it does appear that disallowing type confusion is modestly better than allowing it, and among functional languages, static typing is also somewhat better than dynamic typing. We also find that functional languages are somewhat better than procedural languages. It is worth noting that these modest effects arising from language design are overwhelmingly dominated by the process factors such as project size, team size, and commit size. However, we caution the reader that even these modest effects might quite possibly be due to other, intangible process factors, for example, the preference of certain personality types for functional, static languages that disallow type confusion.

1. INTRODUCTION

A variety of debates ensue during discussions whether a given programming language is “the right tool for the job.” While some of these debates may appear to be tinged with an almost religious fervor, most agree that programming language choice can impact both the coding process and the resulting artifact.

Advocates of strong, static typing tend to believe that the static approach catches defects early; for them, an ounce of prevention is worth a pound of cure. Dynamic typing advocates argue, however, that conservative static type checking is wasteful of developer resources, and that it is better to rely on strong dynamic type checking to catch type errors as they arise. These debates, however, have largely been of the arm-chair variety, supported only by anecdotal evidence.

This is perhaps not unreasonable; obtaining empirical evidence to support such claims is a challenging task given the number of other factors that influence software engineering

outcomes, such as code quality, language properties, and usage domains. Considering, for example, software quality, there are a number of well-known influential factors, such as code size,⁶ team size,² and age/maturity.⁹

Controlled experiments are one approach to examining the impact of language choice in the face of such daunting confounds, however, owing to cost, such studies typically introduce a confound of their own, that is, limited scope. The tasks completed in such studies are necessarily limited and do not emulate *real world* development. There have been several such studies recently that use students, or compare languages with static or dynamic typing through an experimental factor.^{7, 12, 15}

Fortunately, we can now study these questions over a large body of real-world software projects. GitHub contains many projects in multiple languages that substantially vary across size, age, and number of developers. Each project repository provides a detailed record, including contribution history, project size, authorship, and defect repair. We then use a variety of tools to study the effects of language features on defect occurrence. Our approach is best described as mixed-methods, or triangulation⁵ approach; we use text analysis, clustering, and visualization to confirm and support the findings of a quantitative regression study. This empirical approach helps us to understand the practical impact of programming languages, as they are used colloquially by developers, on software quality.

2. METHODOLOGY

Our methods are typical of large scale observational studies in software engineering. We first gather our data from several sources using largely automated methods. We then filter and clean the data in preparation for building a statistical model. We further validate the model using qualitative methods. Filtering choices are driven by a combination of factors including the nature of our research questions, the quality of the data and beliefs about which data is most suitable for statistical study. In particular, GitHub contains many projects written in a large number of programming languages. For this study, we focused our data collection efforts on the most popular projects written in the most popular languages. We choose statistical methods appropriate for evaluating the impact of factors on count data.

The original version of the paper was published in the *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 155–165.

2.1. Data collection

We choose the top 19 programming languages from GitHub. We disregard CSS, Shell script, and Vim script as they are not considered to be general purpose languages. We further include TypeScript, a typed superset of JavaScript. Then, for each of the studied languages we retrieve the top 50 projects that are primarily written in that language. In total, we analyze 850 projects spanning 17 different languages.

Our language and project data was extracted from the *GitHub Archive*, a database that records all public GitHub activities. The archive logs 18 different GitHub events including new commits, fork events, pull request, developers' information, and issue tracking of all the open source GitHub projects on an hourly basis. The archive data is uploaded to Google BigQuery to provide an interface for interactive data analysis.

Identifying top languages. We aggregate projects based on their primary language. Then we select the languages with the most projects for further analysis, as shown in Table 1. A given project can use many languages; assigning a single language to it is difficult. Github Archive stores information gathered from GitHub Linguist which measures the language distribution of a project repository using the source file extensions. The language with the maximum number of source files is assigned as the *primary language* of the project.

Retrieving popular projects. For each selected language, we filter the project repositories written primarily in that language by its popularity based on the associated number of *stars*. This number indicates how many people have actively expressed interest in the project, and is a reasonable proxy for its popularity. Thus, the top 3 projects in C are *linux*, *git*, and *php-src*; and for C++ they are *node-webrtc*, *phantomjs*, and *mongo*; and for Java they are *storm*, *elasticsearch*, and *ActionBarSherlock*. In total, we select the top 50 projects in each language.

To ensure that these projects have a sufficient development history, we drop the projects with fewer than 28 commits (28 is the first quartile commit count of considered

projects). This leaves us with 728 projects. Table 1 shows the top 3 projects in each language.

Retrieving project evolution history. For each of 728 projects, we downloaded the non-merged commits, commit logs, author date, and author name using *git*. We compute code churn and the number of files modified per commit from the number of added and deleted lines per file. We retrieve the languages associated with each commit from the extensions of the modified files (a commit can have multiple language tags). For each commit, we calculate its *commit age* by subtracting its commit date from the first commit of the corresponding project. We also calculate other project-related statistics, including maximum commit age of a project and the total number of developers, used as control variables in our regression model, and discussed in Section 3. We identify bug fix commits made to individual projects by searching for error related keywords: "error," "bug," "fix," "issue," "mistake," "incorrect," "fault," "defect," and "flaw," in the commit log, similar to a prior study.¹⁸

Table 2 summarizes our data set. Since a project may use multiple languages, the second column of the table shows the total number of projects that use a certain language at some capacity. We further exclude some languages from a project that have fewer than 20 commits in that language, where 20 is the first quartile value of the total number of commits per project per language. For example, we find 220 projects that use more than 20 commits in C. This ensures sufficient activity for each language-project pair.

In summary, we study 728 projects developed in 17 languages with 18 years of history. This includes 29,000 different developers, 1.57 million commits, and 564,625 bug fix commits.

2.2. Categorizing languages

We define language classes based on several properties of the language thought to influence language quality,^{7,8,12} as shown in Table 3. The *Programming Paradigm* indicates whether the

Table 1. Top 3 projects in each language.

Language	Projects
C	Linux, git, php-src
C++	Node-webrtc, phantomjs, mongo
C#	SignalR, SparkleShare, ServiceStack
Objective-C	AFNetworking, GPUImage, RestKit
Go	Docker, lime, websocketd
Java	Storm, elasticsearch, ActionBarSherlock
CoffeeScript	Coffee-script, hubot, brunch
JavaScript	Bootstrap, jquery, node
TypeScript	Typescript-node-definitions, StateTree, typescript.api
Ruby	Rails, gitlabhq, homebrew
Php	Laravel, CodeIgniter, symfony
Python	Flask, django, reddit
Perl	Gitolite, showdown, rails-dev-box
Clojure	LightTable, leiningen, clojurescript
Erlang	ChicagoBoss, cowboy, couchdb
Haskell	Pandoc, yesod, git-annex
Scala	Play20, spark, scala

Table 2. Study subjects.

Language	Project details		Commits		
	#Projects	#Devs (K)	#Commits (K)	#Insertion (MLOC)	#BugFixes (K)
C	220	13.8	447.8	75.3	182.6
C++	149	3.8	196.5	46.0	79.3
C#	77	2.3	135.8	27.7	50.7
Objective-C	93	1.6	21.6	2.4	7.1
Go	54	6.6	19.7	1.6	4.4
Java	141	3.3	87.1	19.1	35.1
CoffeeScript	92	1.7	22.5	1.1	6.3
JavaScript	432	6.8	118.3	33.1	39.3
TypeScript	14	2.4	3.3	2.0	0.9
Ruby	188	9.6	122.1	5.8	30.5
Php	109	4.9	118.7	16.2	47.2
Python	286	5.0	114.2	9.0	41.9
Perl	106	0.8	5.5	0.5	1.9
Clojure	60	0.8	28.4	1.5	6.0
Erlang	51	0.8	31.4	5.0	8.1
Haskell	55	0.9	46.1	2.9	10.4
Scala	55	1.3	55.7	5.3	12.9
Summary	728	28	1574	254	564

Table 3. Different types of language classes.

Language classes	Categories	Languages
Programming paradigm	Imperative procedural	C, C++, C#, Objective-C, Java, Go
	Imperative scripting	CoffeeScript, JavaScript, Python, Perl, Php, Ruby
	Functional	Clojure, Erlang, Haskell, Scala
Type checking	Static	C, C++, C#, Objective-C, Java, Go, Haskell, Scala
	Dynamic	CoffeeScript, JavaScript, Python, Perl, Php, Ruby, Clojure, Erlang
Implicit type conversion	Disallow	C#, Java, Go, Python, Ruby, Clojure, Erlang, Haskell, Scala
	Allow	C, C++, Objective-C, CoffeeScript, JavaScript, Perl, Php
Memory class	Managed	Others
	Unmanaged	C, C++, Objective-C

We omit TypeScript from language classification as it allows both explicit and implicit type conversion.

project is written in an imperative procedural, imperative scripting, or functional language. In the rest of the paper, we use the terms procedural and scripting to indicate imperative procedural and imperative scripting respectively.

Type Checking indicates static or dynamic typing. In statically typed languages, type checking occurs at compile time, and variable names are bound to a value and to a type. In addition, expressions (including variables) are classified by types that correspond to the values they might take on at run-time. In dynamically typed languages, type checking occurs at run-time. Hence, in the latter, it is possible to bind a variable name to objects of different types in the same program.

Implicit Type Conversion allows access of an operand of type T1 as a different type T2, without an explicit conversion. Such implicit conversion may introduce type-confusion in some cases, especially when it presents an operand of specific type T1, as an instance of a different type T2. Since not all implicit type conversions are immediately a problem, we operationalize our definition by showing examples of the implicit type confusion that can happen in all the languages we identified as allowing it. For example, in languages like Perl, JavaScript, and CoffeeScript adding a string to a number is permissible (e.g., “5” + 2 yields “52”). The same operation yields 7 in Php. Such an operation is not permitted in languages such as Java and Python as they do not allow implicit conversion. In C and C++ coercion of data types can result in unintended results, for example, `int x; float y; y=3.5; x=y;` is legal C code, and results in different values for x and y, which, depending on intent, may be a problem downstream.^a In Objective-C the data type *id* is a generic object pointer, which can be used with an object of any data type, regardless of the class.^b The flexibility that such a generic

data type provides can lead to implicit type conversion and also have unintended consequences.^c Hence, we classify a language based on whether its compiler *allows* or *disallows* the implicit type conversion as above; the latter explicitly detects type confusion and reports it.

Disallowing implicit type conversion could result from static type inference within a compiler (e.g., with Java), using a type-inference algorithm such as Hindley¹⁰ and Milner,¹⁷ or at run-time using a dynamic type checker. In contrast, a type-confusion can occur silently because it is either undetected or is unreported. Either way, implicitly allowing type conversion provides flexibility but may eventually cause errors that are difficult to localize. To abbreviate, we refer to languages allowing implicit type conversion as *implicit* and those that disallow it as *explicit*.

Memory Class indicates whether the language requires developers to manage memory. We treat Objective-C as unmanaged, in spite of it following a hybrid model, because we observe many memory errors in its codebase, as discussed in RQ4 in Section 3.

Note that we classify and study the languages as they are colloquially used by developers in real-world software. For example, TypeScript is intended to be used as a static language, which disallows implicit type conversion. However, in practice, we notice that developers often (for 50% of the variables, and across TypeScript-using projects in our dataset) use the any type, a catch-all union type, and thus, in practice, TypeScript allows dynamic, implicit type conversion. To minimize the confusion, we exclude TypeScript from our language classifications and the corresponding model (see Table 3 and 7).

2.3. Identifying project domain

We classify the studied projects into different domains based on their features and function using a mix of automated and manual techniques. The projects in GitHub come with project descriptions and README files that describe their features. We used Latent Dirichlet Allocation (LDA)³ to analyze this text. Given a set of documents, LDA identifies a set of topics where each topic is represented as probability of generating different words. For each document, LDA also estimates the probability of assigning that document to each topic.

We detect 30 distinct domains, that is, topics, and estimate the probability that each project belonging to each domain. Since these auto-detected domains include several project-specific keywords, for example, facebook, it is difficult to identify the underlying common functions. In order to assign a meaningful name to each domain, we manually inspect each of the 30 domains to identify projectname-independent, domain-identifying keywords. We manually rename all of the 30 auto-detected domains and find that the majority of the projects fall under six domains: Application, Database, CodeAnalyzer, Middleware, Library, and Framework. We also find that some projects do not fall under any of the above

^a Wikipedia’s article on type conversion, https://en.wikipedia.org/wiki/Type_conversion, has more examples of unintended behavior in C.

^b This Apple developer article describes the usage of “id” <http://tinyurl.com/jkl7cby>.

^c Some examples can be found here <http://dobegin.com/objc-id-type/> and here <http://tinyurl.com/hxv8kvq>.

domains and so we assign them to a catchall domain labeled as *Other*. This classification of projects into domains was subsequently checked and confirmed by another member of our research group. Table 4 summarizes the identified domains resulting from this process.

2.4. Categorizing bugs

While fixing software bugs, developers often leave important information in the commit logs about the nature of the bugs; for example, why the bugs arise and how to fix the bugs. We exploit such information to categorize the bugs, similar to Tan *et al.*^{13,24}

First, we categorize the bugs based on their *Cause* and *Impact*. *Causes* are further classified into disjoint subcategories of errors: Algorithmic, Concurrency, Memory, generic Programming, and Unknown. The bug *Impact* is also classified into four disjoint subcategories: Security, Performance, Failure, and Other unknown categories. Thus, each bug-fix commit also has an induced Cause and an Impact type. Table 5 shows the description of each bug category. This classification is performed in two phases:

(1) **Keyword search.** We randomly choose 10% of the bug-fix messages and use a keyword based search technique to automatically categorize them as potential bug types. We use this annotation, separately, for both Cause and Impact types. We chose a restrictive set of keywords and phrases, as shown in Table 5. Such a restrictive set of keywords and phrases helps reduce false positives.

Table 4. Characteristics of domains.

Domain name	Domain characteristics	Example projects	Total projects
(APP) Application	End user programs	bitcoin, macvim	120
(DB) Database	SQL and NoSQL	mysql, mongodb	43
(CA) CodeAnalyzer	Compiler, parser, etc.	ruby, php-src	88
(MW) Middleware	OS, VMs, etc.	linux, memcached	48
(LIB) Library	APIs, libraries, etc.	androidApis, opencv	175
(FW) Framework	SDKs, plugins	ios sdk, coffeekup	206
(OTH) Other	–	Arduino, autoenv	49

(2) **Supervised classification.** We use the annotated bug fix logs from the previous step as training data for supervised learning techniques to classify the remainder of the bug fix messages by treating them as test data. We first convert each bug fix message to a bag-of- words. We then remove words that appear only once among all of the bug fix messages. This reduces project specific keywords. We also stem the bag-of- words using standard natural language processing techniques. Finally, we use Support Vector Machine to classify the test data.

To evaluate the accuracy of the bug classifier, we manually annotated 180 randomly chosen bug fixes, equally distributed across all of the categories. We then compare the result of the automatic classifier with the manually annotated data set. The performance of this process was acceptable with precision ranging from a low of 70% for performance bugs to a high of 100% for concurrency bugs with an average of 84%. Recall ranged from 69% to 91% with an average of 84%.

The result of our bug classification is shown in Table 5. Most of the defect causes are related to generic programming errors. This is not surprising as this category involves a wide variety of programming errors such as type errors, typos, compilation error, etc. Our technique could not classify 1.04% of the bug fix messages in any Cause or Impact category; we classify these as Unknown.

2.5. Statistical methods

We model the number of defective commits against other factors related to software projects using regression. All models use *negative binomial regression* (NBR) to model the counts of project attributes such as the number of commits. NBR is a type of generalized linear model used to model non-negative integer responses.⁴

In our models we control for several language per-project dependent factors that are likely to influence the outcome. Consequently, each (language, project) pair is a row in our regression and is viewed as a sample from the population of open source projects. We log-transform dependent count variables as it stabilizes the variance and usually improves the model fit.⁴ We verify this by comparing transformed with non transformed data using the AIC and Vuong's test for non-nested models.

Table 5. Categories of bugs and their distribution in the whole dataset.

	Bug type	Bug description	Search keywords/phrases	Count	% count
Cause	Algorithm (Algo)	Algorithmic or logical errors	Algorithm	606	0.11
	Concurrency (Conc)	Multithreading/processing issues	Deadlock, race condition, synchronization error	11,111	1.99
	Memory (Mem)	Incorrect memory handling	Memory leak, null pointer, buffer overflow, heap overflow, null pointer, dangling pointer, double free, segmentation fault	30,437	5.44
	Programming (Prog)	Generic programming errors	Exception handling, error handling, type error, typo, compilation error, copy-paste error, refactoring, missing switch case, faulty initialization, default value	495,013	88.53
Impact	Security (Sec)	Runs, but can be exploited	Buffer overflow, security, password, oauth, ssl	11,235	2.01
	Performance (Perf)	Runs, but with delayed response	Optimization problem, performance	8651	1.55
	Failure (Fail)	Crash or hang	Reboot, crash, hang, restart	21,079	3.77
	Unknown (Unkn)	Not part of the above categories		5792	1.04

To check that excessive multicollinearity is not an issue, we compute the variance inflation factor of each dependent variable in all of the models with a conservative maximum value of 5.⁴ We check for and remove high leverage points through visual examination of the residuals versus leverage plot for each model, looking for both separation and large values of Cook's distance.

We employ *effects*, or *contrast*, coding in our study to facilitate interpretation of the language coefficients.⁴ Weighted effects codes allow us to compare each language to the average effect across all languages while compensating for the unevenness of language usage across projects.²³ To test for the relationship between two factor variables we use a Chi-square test of independence.¹⁴ After confirming a dependence we use Cramer's V, an $r \times c$ equivalent of the phi coefficient for nominal data, to establish an effect size.

3. RESULTS

We begin with a straightforward question that directly addresses the core of what some fervently believe must be true, namely:

RQ1. Are some languages more defect-prone than others?

We use a regression model to compare the impact of each language on the number of defects with the average impact of all languages, against defect fixing commits (see Table 6).

We include some variables as controls for factors that will clearly influence the response. Project age is included as older projects will generally have a greater number of defect fixes. Trivially, the number of commits to a project will also impact the response. Additionally, the number of developers who touch a project and the raw size of the project are both expected to grow with project activity.

Table 6. Some languages induce fewer defects than other languages.

Defective commits model	Coeff. (Std. Err.)
(Intercept)	-2.04 (0.11)***
Log age	0.06 (0.02)***
Log size	0.04 (0.01)***
Log devs	0.06 (0.01)***
Log commits	0.96 (0.01)***
C	0.11 (0.04)**
C++	0.18 (0.04)***
C#	-0.02 (0.05)
Objective-C	0.15 (0.05)**
Go	-0.11 (0.06)
Java	-0.06 (0.04)
CoffeeScript	0.06 (0.05)
JavaScript	0.03 (0.03)
TypeScript	0.15 (0.10)
Ruby	-0.13 (0.05)**
Php	0.10 (0.05)*
Python	0.08 (0.04)*
Perl	-0.12 (0.08)
Clojure	-0.30 (0.05)***
Erlang	-0.03 (0.05)
Haskell	-0.26 (0.06)***
Scala	-0.24 (0.05)***

Response is the number of defective commits. Languages are coded with weighted effects coding. AIC=10432, Deviance=1156, Num. obs.=1076.
 *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

The sign and magnitude of the estimated coefficients in the above model relates the predictors to the outcome. The first four variables are control variables and we are not interested in their impact on the outcome other than to say that they are all positive and significant. The language variables are indicator variables, viz. factor variables, for each project. The coefficient compares each language to the grand weighted mean of all languages in all projects. The language coefficients can be broadly grouped into three general categories. The first category is those for which the coefficient is statistically insignificant and the modeling procedure could not distinguish the coefficient from zero. These languages may behave similar to the average or they may have wide variance. The remaining coefficients are significant and either positive or negative. For those with positive coefficients we can expect that the language is associated with a greater number of defect fixes. These languages include C, C++, Objective-C, Php, and Python. The languages Clojure, Haskell, Ruby, and Scala, all have negative coefficients implying that these languages are less likely than average to result in defect fixing commits.

One should take care not to overestimate the impact of language on defects. While the observed relationships are statistically significant, the effects are quite small. Analysis of deviance reveals that language accounts for less than 1% of the total explained deviance.

	Df	Deviance	Resid. Df	Resid. dev	Pr (>Chi)
NULL			1075	25,176.25	
Log commits	1	4256.89	1071	1286.74	0
Log age	1	8011.52	1074	17,164.73	0
Log size	1	10,082.78	1073	7081.95	0
Log devs	1	1538.32	1072	5543.63	0
Language	16	130.78	1055	1155.96	0

We can read the model coefficients as the expected change in the log of the response for a one unit change in the predictor with all other predictors held constant; that is, for a coefficient β_i , a one unit change in β_i yields an expected change in the response of e^{β_i} . For the factor variables, this expected change is compared to the average across all languages. Thus, if, for some number of commits, a particular project developed in an *average* language had four defective commits, then the choice to use C++ would mean that we should expect one additional defective commit since $e^{0.18} \times 4 = 4.79$. For the same project, choosing Haskell would mean that we should expect about one fewer defective commit as $e^{-0.26} \times 4 = 3.08$. The accuracy of this prediction depends on all other factors remaining the same, a challenging proposition for all but the most trivial of projects. All observational studies face similar limitations; we address this concern in more detail in Section 5.

Result 1: *Some languages have a greater association with defects than other languages, although the effect is small.*

In the remainder of this paper we expand on this basic result by considering how different categories of application, defect, and language, lead to further insight into the relationship between languages and defect proneness.

Software bugs usually fall under two broad categories: (1) *Domain Specific bug*: specific to project function and do not

depend on the underlying programming language. (2) *Generic bug*: more generic in nature and has less to do with project function, for example, typeerrors, concurrency errors, etc.

Consequently, it is reasonable to think that the interaction of application domain and language might impact the number of defects within a project. Since some languages are believed to excel at some tasks more so than others, for example, C for low level work, or Java for user applications, making an inappropriate choice might lead to a greater number of defects. To study this we should ideally ignore the domain specific bugs, as generic bugs are more likely to depend on the programming language featured. However, since a domain-specific bugs may also arise due to a generic programming error, it is difficult to separate the two. A possible workaround is to study languages while controlling the domain. Statistically, however, with 17 languages across 7 domains, the large number of terms would be challenging to interpret given the sample size.

Given this, we first consider testing for the dependence between domain and language usage within a project, using a Chi-square test of independence. Of 119 cells, 46, that is, 39%, are below the value of 5 which is too high. No more than 20% of the counts should be below 5.¹⁴ We include the value here for completeness^d; however, the low strength of association of 0.191 as measured by Cramer's V, suggests that any relationship between domain and language is small and that inclusion of domain in regression models would not produce meaningful results.

One option to address this concern would be to remove languages or combine domains, however, our data here presents no clear choices. Alternatively, we could combine languages; this choice leads to a related but slightly different question.

RQ2. Which language properties relate to defects?

Rather than considering languages individually, we aggregate them by language class, as described in Section 2.2, and analyze the relationship to defects. Broadly, each of these properties divides languages along lines that are often discussed in the context of errors, drives user debate, or has been the subject of prior work. Since the individual properties are highly correlated, we create six model factors that combine all of the individual factors across all of the languages in our study. We then model the impact of the six different factors on the number of defects while controlling for the same basic covariates that we used in the model in RQ1.

As with language (earlier in Table 6), we are comparing language *classes* with the average behavior across all language classes. The model is presented in Table 7. It is clear that Script-Dynamic-Explicit-Managed class has the smallest magnitude coefficient. The coefficient is insignificant, that is, the z-test for the coefficient cannot distinguish the coefficient from zero. Given the magnitude of the standard error, however, we can assume that the behavior of languages in this class is very close to the average across all languages. We confirm this by recoding the coefficient using Proc-Static-Implicit-Unmanaged as the base level and employing treatment, or dummy coding that compares each language class with the base level. In this case, Script-Dynamic-Explicit-Managed is significantly

^d Chi-squared value of 243.6 with 96 *df.* and $p = 8.394e-15$

Table 7. Functional languages have a smaller relationship to defects than other language classes whereas procedural languages are greater than or similar to the average.

Defective commits	
(Intercept)	-2.13 (0.10)***
Log commits	0.96 (0.01)***
Log age	0.07 (0.01)***
Log size	0.05 (0.01)***
Log devs	0.07 (0.01)***
Functional-Static-Explicit-Managed	-0.25 (0.04)***
Functional-Dynamic-Explicit-Managed	-0.17 (0.04)***
Proc-Static-Explicit-Managed	-0.06 (0.03)*
Script-Dynamic-Explicit-Managed	0.001 (0.03)
Script-Dynamic-Implicit-Managed	0.04 (0.02)*
Proc-Static-Implicit-Unmanaged	0.14 (0.02)***

Language classes coded with weighted effects codes (AIC = 10,419, Deviance = 1132, Num. obs. = 1067).
*** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

different with $p = 0.00044$. We note here that while choosing different coding methods affects the coefficients and z-scores, the models are identical in all other respects. When we change the coding we are rescaling the coefficients to reflect the comparison that we wish to make.⁴ Comparing the other language classes to the grand mean, Proc-Static-Implicit-Unmanaged languages are more likely to induce defects. This implies that either implicit type conversion or memory management issues contribute to greater defect proneness as compared with other procedural languages.

Among scripting languages we observe a similar relationship between languages that allow versus those that do not allow implicit type conversion, providing some evidence that implicit type conversion (vs. explicit) is responsible for this difference as opposed to memory management. We cannot state this conclusively given the correlation between factors. However when compared to the average, as a group, languages that do not allow implicit type conversion are less error-prone while those that do are more error-prone. The contrast between static and dynamic typing is also visible in functional languages.

The functional languages as a group show a strong difference from the average. Statically typed languages have a substantially smaller coefficient yet both functional language classes have the same standard error. This is strong evidence that functional static languages are less error-prone than functional dynamic languages, however, the z-tests only test whether the coefficients are different from zero. In order to strengthen this assertion, we recode the model as above using treatment coding and observe that the Functional-Static-Explicit-Managed language class is significantly less defect-prone than the Functional-Dynamic-Explicit-Managed language class with $p = 0.034$.

	Df	Deviance	Resid. Df	Resid. Dev	Pr (>Chi)
NULL			1066	32,995.23	
Log commits	1	31,634.32	1065	1360.91	0
Log age	1	51.04	1064	1309.87	0
Log size	1	50.82	1063	1259.05	0
Log devs	1	31.11	1062	1227.94	0
Lang. class	5	95.54	1057	1132.40	0

As with language and defects, the relationship between language class and defects is based on a small effect. The deviance explained is similar, albeit smaller, with language class explaining much less than 1% of the deviance.

We now revisit the question of application domain. Does domain have an interaction with language class? Does the choice of, for example, a functional language, have an advantage for a particular domain? As above, a Chi-square test for the relationship between these factors and the project domain yields a value of 99.05 and $df = 30$ with $p = 2.622e-09$ allowing us to reject the null hypothesis that the factors are independent. Cramer's-V yields a value of 0.133, a weak level of association. Consequently, although there is some relation between domain and language, there is only a weak relationship between domain and language class.

Result 2: *There is a small but significant relationship between language class and defects. Functional languages are associated with fewer defects than either procedural or scripting languages.*

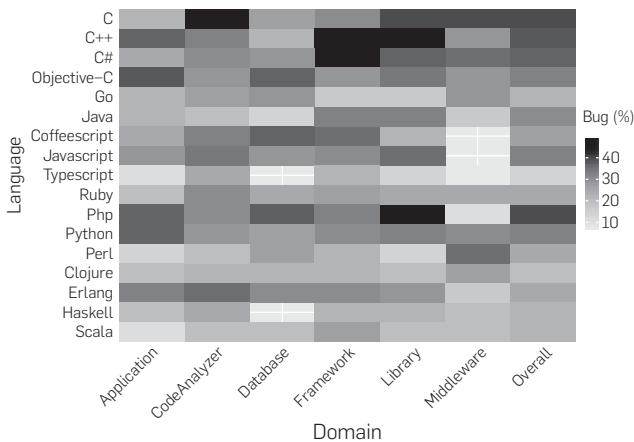
It is somewhat unsatisfying that we do not observe a strong association between language, or language class, and domain within a project. An alternative way to view this same data is to disregard projects and aggregate defects over all languages and domains. Since this does not yield independent samples, we do not attempt to analyze it statistically, rather we take a descriptive, visualization-based approach.

We define *Defect Proneness* as the ratio of bug fix commits over total commits per language per domain. Figure 1 illustrates the interaction between domain and language using a heat map, where the defect proneness increases from lighter to darker zone. We investigate which language factors influence defect fixing commits across a collection of projects written across a variety of languages. This leads to the following research question:

RQ3. Does language defect proneness depend on domain?

In order to answer this question we first filtered out projects that would have been viewed as outliers, filtered as high leverage points, in our regression models. This was necessary

Figure 1. Interaction of language's defect proneness with domain. Each cell in the heat map represents defect proneness of a language (row header) for a given domain (column header). The "Overall" column represents defect proneness of a language over all the domains. The cells with white cross mark indicate null value, that is, no commits were made corresponding to that cell.



here as, even though this is a nonstatistical method, some relationships could impact visualization. For example, we found that a single project, Google's v8, a JavaScript project, was responsible for all of the errors in Middleware. This was surprising to us since JavaScript is typically not used for Middleware. This pattern repeats in other domains, consequently, we filter out the projects that have defect density below 10 and above 90 percentile. The result is in Figure 1.

We see only a subdued variation in this heat map which is a result of the inherent defect proneness of the languages as seen in RQ1. To validate this, we measure the pairwise rank correlation between the language defect proneness for each domain with the overall. For all of the domains except Database, the correlation is positive, and p-values are significant (<0.01). Thus, w.r.t. defect proneness, the language ordering in each domain is strongly correlated with the overall language ordering.

	APP	CA	DB	FW	LIB	MW
Spearman corr.	0.71	0.56	0.30	0.76	0.90	0.46
p-Value	0.00	0.02	0.28	0.00	0.00	0.09

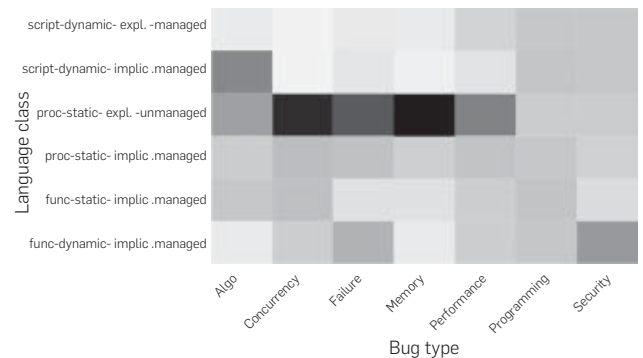
Result 3: *There is no general relationship between application domain and language defect proneness.*

We have shown that different languages induce a larger number of defects and that this relationship is not only related to particular languages but holds for general classes of languages; however, we find that the type of project does not mediate this relationship to a large degree. We now turn our attention to categorization of the response. We want to understand how language relates to specific kinds of defects and how this relationship compares to the more general relationship that we observe. We divide the defects into categories as described in Table 5 and ask the following question:

RQ4. What is the relation between language and bug category?

We use an approach similar to RQ3 to understand the relation between languages and bug categories. First, we study the relation between bug categories and language class. A heat map (Figure 2) shows aggregated defects over language classes and bug types. To understand the interaction between

Figure 2. Relation between bug categories and language class. Each cell represents percentage of bug fix commit out of all bug fix commits per language class (row header) per bug category (column header). The values are normalized column wise.



bug categories and languages, we use an NBR regression model for each category. For each model we use the same control factors as RQ1 as well as languages encoded with weighted effects to predict defect fixing commits.

The results along with the anova value for language are shown in Table 8. The overall deviance for each model is substantially smaller and the proportion explained by language for a specific defect type is similar in magnitude for most of the categories. We interpret this relationship to mean that language has a greater impact on specific categories of bugs, than it does on bugs overall. In the next section we expand on these results for the bug categories with significant bug counts as reported in Table 5. However, our conclusion generalizes for all categories.

Programming errors. Generic programming errors account for around 88.53% of all bug fix commits and occur in all the language classes. Consequently, the regression analysis draws a similar conclusion as of RQ1 (see Table 6). All languages incur programming errors such as faulty error-handling, faulty definitions, typos, etc.

Memory errors. Memory errors account for 5.44% of all the bug fix commits. The heat map in Figure 2 shows a strong relationship between Proc-Static-Implicit-Unmanaged class and memory errors. This is expected as languages with unmanaged memory are known for memory bugs. Table 8 confirms that such languages, for example, C, C++, and Objective-C introduce more memory errors. Among the

managed languages, Java induces more memory errors, although fewer than the unmanaged languages. Although Java has its own garbage collector, memory leaks are not surprising since unused object references often prevent the garbage collector from reclaiming memory.¹¹ In our data, 28.89% of all the memory errors in Java are the result of a memory leak. In terms of effect size, language has a larger impact on memory defects than all other *cause* categories.

Concurrency errors. 1.99% of the total bug fix commits are related to concurrency errors. The heat map shows that Proc-Static-Implicit-Unmanaged dominates this error type. C and C++ introduce 19.15% and 7.89% of the errors, and they are distributed across the projects.

	C	C++	C#	Java	Scala	Go	Erlang
Race	63.11	41.46	77.7	65.35	74.07	92.08	78.26
Deadlock	26.55	43.36	14.39	17.08	18.52	10.89	15.94
SHM	28.78	18.24	9.36	9.16	8.02	0	0
MPI	0	2.21	2.16	3.71	4.94	1.98	10.14

Both of the Static-Strong-Managed language classes are in the darker zone in the heat map confirming, in general static languages produce more concurrency errors than others. Among the dynamic languages, only Erlang is more prone to concurrency errors, perhaps relating to the greater use of this language for concurrent applications. Likewise, the negative coefficients in Table 8 shows that projects written

Table 8. While the impact of language on defects varies across defect category, language has a greater impact on specific categories than it does on defects in general.

	Memory	Concurrency	Security	Failure
(Intercept)	-7.49 (0.46)***	-8.13 (0.74)***	-7.29 (0.58)***	-6.21 (0.41)***
Log commits	0.99 (0.05)***	1.09 (0.09)***	0.89 (0.07)***	0.88 (0.05)***
Log age	0.15 (0.06)*	0.19 (0.10)	0.30 (0.08)***	0.07 (0.06)
Log size	0.01 (0.04)	-0.08 (0.07)	-0.01 (0.05)	0.14 (0.04)***
Log devs	0.07 (0.04)	0.09 (0.07)	0.07 (0.06)	-0.11 (0.04)*
C	1.71 (0.12)***	0.39 (0.22)	0.28 (0.18)	0.43 (0.13)**
C#	-0.12 (0.17)	0.81 (0.24)***	-0.42 (0.23)	-0.07 (0.16)
C++	1.08 (0.10)***	1.07 (0.18)***	0.40 (0.16)*	1.05 (0.11)***
Objective-C	1.40 (0.15)***	0.41 (0.28)	-0.14 (0.24)	1.10 (0.15)***
Go	-0.05 (0.25)	1.62 (0.30)***	0.35 (0.28)	-0.49 (0.24)*
Java	0.53 (0.14)***	0.80 (0.22)***	-0.07 (0.19)	0.15 (0.14)
CoffeeScript	-0.41 (0.23)	-1.73 (0.54)**	-0.36 (0.27)	-0.05 (0.19)
JavaScript	-0.16 (0.10)	-0.21 (0.16)	0.02 (0.12)	-0.15 (0.09)
TypeScript	-0.58 (0.62)	-0.63 (1.02)	0.37 (0.51)	-0.42 (0.41)
Ruby	-1.16 (0.19)***	-0.89 (0.29)**	-0.18 (0.21)	-0.32 (0.16)*
Php	-0.69 (0.17)***	-1.70 (0.34)***	0.11 (0.21)	-0.62 (0.17)***
Python	-0.48 (0.14)***	-0.25 (0.22)	0.36 (0.16)*	0.04 (0.12)
Perl	0.15 (0.35)	-1.23 (0.83)	-0.62 (0.45)	-0.64 (0.38)
Scala	-0.47 (0.18)**	0.63 (0.24)**	-0.22 (0.22)	-0.93 (0.18)***
Clojure	-1.21 (0.27)***	-0.01 (0.30)	-0.82 (0.27)**	-0.62 (0.19)**
Erlang	-0.60 (0.23)**	0.63 (0.28)*	0.62 (0.22)**	0.59 (0.17)***
Haskell	-0.28 (0.20)	-0.27 (0.32)	-0.45 (0.26)	-0.49 (0.20)*
AIC	2991.47	2210.01	3328.39	4086.42
Deviance	895.02	665.17	896.58	1043.02
Num. obs.	1081	1081	1073	1077
Residual deviance (NULL)	5065.30	2124.93	2170.23	3769.70
Language deviance	522.86	139.67	42.72	240.51

For all models the deviance explained by language type has $p < 0.0003076$.
 *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$.

in dynamic languages like Ruby and Php have fewer concurrency errors. Note that, certain languages like JavaScript, CoffeeScript, and TypeScript do not support concurrency, in its traditional form, while Php has a limited support depending on its implementations. These languages introduce artificial zeros in the data, and thus the concurrency model coefficients in Table 8 for those languages cannot be interpreted like the other coefficients. Due to these artificial zeros, the average over all languages in this model is smaller, which may affect the sizes of the coefficients, since they are given w.r.t. the average, but it will not affect their relative relationships, which is what we are after.

A textual analysis based on word-frequency of the bug fix messages suggests that most of the concurrency errors occur due to a race condition, deadlock, or incorrect synchronization, as shown in the table above. Across all language, race conditions are the most frequent cause of such errors, for example, 92% in Go. The enrichment of race condition errors in Go is probably due to an accompanying race-detection tool that may help developers locate races. The synchronization errors are primarily related to message passing interface (MPI) or shared memory operation (SHM). Erlang and Go use MPI^c for inter-thread communication, which explains why these two languages do not have any SHM related errors such as locking, mutex, etc. In contrast, projects in the other languages use SHM primitives for communication and can thus may have locking-related errors.

Security and other impact errors. Around 7.33% of all the bug fix commits are related to Impact errors. Among them Erlang, C++, and Python associate with more security errors than average (Table 8). Clojure projects associate with fewer security errors (Figure 2). From the heat map we also see that Static languages are in general more prone to failure and performance errors, these are followed by Functional-Dynamic-Explicit-Managed languages such as Erlang. The analysis of deviance results confirm that language is strongly associated with failure impacts. While security errors are the weakest among the categories, the deviance explained by language is still quite strong when compared with the residual deviance.

Result 4: *Defect types are strongly associated with languages; some defect type like memory errors and concurrency errors also depend on language primitives. Language matters more for specific categories than it does for defects overall.*

4. RELATED WORK

Prior work on programming language comparison falls in three categories:

(1) **Controlled experiment.** For a given task, developers are monitored while programming in different languages. Researchers then compare outcomes such as development effort and program quality. Hanenberg⁷ compared static versus dynamic typing by monitoring 48 programmers for 27 h while developing a parser program. He found no significant difference in code quality between the two; however, dynamic type-based languages were found to have shorter development time. Their study was conducted with undergraduate

students in a lab setting with custom-designed language and IDE. Our study, by contrast is a field study of popular software applications. While we can only indirectly (and *post facto*) control for confounding factors using regression, we benefit from much larger sample sizes, and more realistic, widely-used software. We find that statically typed languages in general are less defect-prone than the dynamic types, and that disallowing implicit type conversion is better than allowing it, in the same regard. The effect sizes are modest; it could be reasonably argued that they are visible here precisely because of the large sample sizes.

Harrison et al.⁸ compared C++, a procedural language, with SML, a functional language, finding no significant difference in total number of errors, although SML has higher defect density than C++. SML was not represented in our data, which however, suggest that functional languages are generally less defect-prone than procedural languages. Another line of work primarily focuses on comparing development effort across different languages.^{12, 20} However, they do not analyze language defect proneness.

(2) **Surveys.** Meyerovich and Rabkin¹⁶ survey developers' views of programming languages, to study why some languages are more popular than others. They report strong influence from non-linguistic factors: prior language skills, availability of open source tools, and existing legacy systems. Our study also confirms that the availability of external tools also impacts software quality; for example, concurrency bugs in Go (see RQ4 in Section 3).

(3) **Repository mining.** Bhattacharya and Neamtiu¹ study four projects developed in both C and C++ and find that the software components developed in C++ are in general more reliable than C. We find that both C and C++ are more defect-prone than average. However, for certain bug types like concurrency errors, C is more defect-prone than C++ (see RQ4 in Section 3).

5. THREATS TO VALIDITY

We recognize few threats to our reported results. First, to identify bug fix commits we rely on the keywords that developers often use to indicate a bug fix. Our choice was deliberate. We wanted to capture the issues that developers continuously face in an ongoing development process, rather than reported bugs. However, this choice possesses threats of over estimation. Our categorization of domains is subject to interpreter bias, although another member of our group verified the categories. Also, our effort to categorize bug fix commits could potentially be tainted by the initial choice of keywords. The descriptiveness of commit logs vary across projects. To mitigate these threats, we evaluate our classification against manual annotation as discussed in Section 2.4.

We determine the language of a file based on its extension. This can be error-prone if a file written in a different language takes a common language extension that we have studied. To reduce such error, we manually verified language categorization against a randomly sampled file set.

To interpret language class in Section 2.2, we make certain assumptions based on how a language property is most commonly used, as reflected in our data set, for example, we classify Objective-C as unmanaged memory type rather

^c MPI does not require locking of shared resources.

than hybrid. Similarly, we annotate Scala as functional and C# as procedural, although both support either design choice.^{19, 21} We do not distinguish object-oriented languages (OOP) from procedural languages in this work as there is no clear distinction, the difference largely depends on programming style. We categorize C++ as allowing implicit type conversion because a memory region of a certain type can be treated differently using pointer manipulation.²² We note that most C++ compilers can detect type errors at compile time.

Finally, we associate defect fixing commits to language properties, although they could reflect reporting style or other developer properties. Availability of external tools or libraries may also impact the extent of bugs associated with a language.

6. CONCLUSION

We have presented a large-scale study of language type and use as it relates to software quality. The Github data we used is characterized by its complexity and variance along multiple dimensions. Our sample size allows a mixed-methods study of the effects of language, and of the interactions of language, domain, and defect type while controlling for a number of confounds. The data indicates that functional languages are better than procedural languages; it suggests that disallowing implicit type conversion is better than allowing it; that static typing is better than dynamic; and that managed memory usage is better than unmanaged. Further, that the defect proneness of languages in general is not associated with software domains. Additionally, languages are more related to individual bug categories than bugs overall.

On the other hand, even large datasets become small and insufficient when they are sliced and diced many ways simultaneously. Consequently, with an increasing number of dependent variables it is difficult to answer questions about a specific variable's effect, especially where variable interactions exist. Hence, we are unable to quantify the specific effects of language type on usage. Additional methods such as surveys could be helpful here. Addressing these challenges remains for future work.

Acknowledgments

This material is based upon work supported by the National Science Foundation under grant nos. 1445079, 1247280, 1414172, 1446683 and from AFOSR award FA955-11-1-0246. 

References

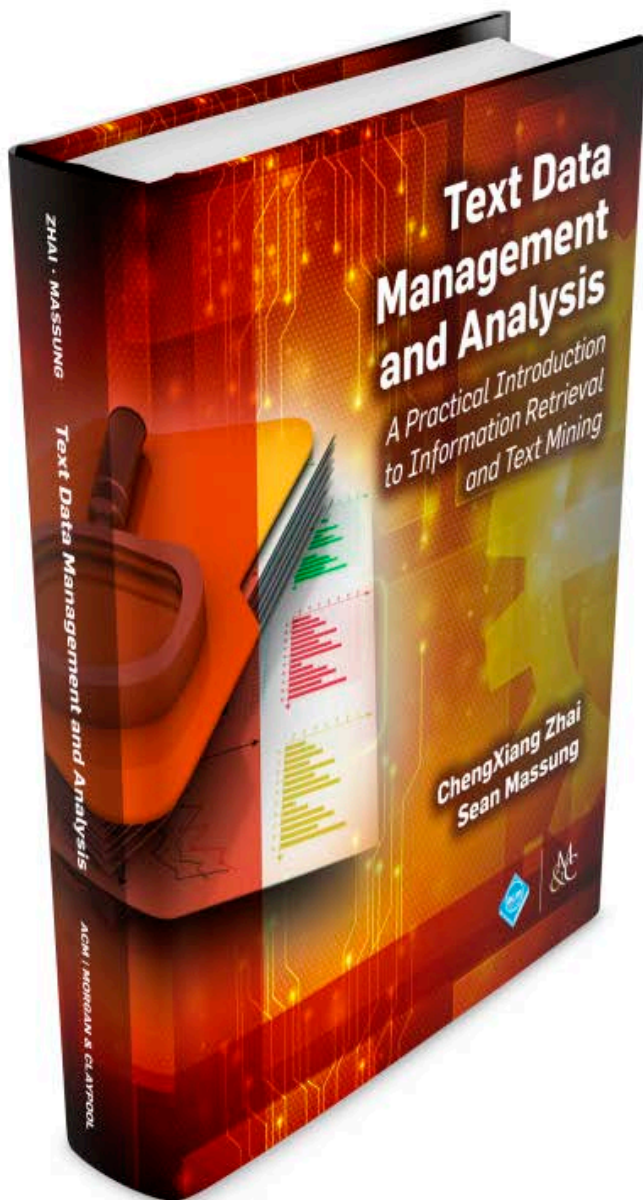
- Bhattacharya, P., Neamtiu, I. Assessing programming language impact on development and maintenance: A study on C and C++. In *Proceedings of the 33rd International Conference on Software Engineering, ICSE'11* (New York, NY, USA, 2011). ACM, 171–180.
- Bird, C., Nagappan, N., Murphy, B., Gall, H., Devanbu, P. Don't touch my code! Examining the effects of ownership on software quality. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering* (2011). ACM, 4–14.
- Blei, D.M. Probabilistic topic models. *Commun. ACM* 55, 4 (2012), 77–84.
- Cohen, J. *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Lawrence Erlbaum, 2003.
- Easterbrook, S., Singer, J., Storey, M.-A., Damian, D. Selecting empirical methods for software engineering research. In *Guide to Advanced Empirical Software Engineering* (2008). Springer, 285–311.
- El Emam, K., Benlarbi, S., Goel, N., Rai, S.N. The confounding effect of class size on the validity of object-oriented metrics. *IEEE Trans. Softw. Eng.* 27, 7 (2001), 630–650.
- Hanenberg, S. An experiment about static and dynamic type systems: Doubts about the positive impact of static type systems on development

- time. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications, OOPSLA'10* (New York, NY, USA, 2010). ACM, 22–35.
- Harrison, R., Smaraweera, L., Dobie, M., Lewis, P. Comparing programming paradigms: An evaluation of functional and object-oriented programs. *Softw. Eng. J.* 11, 4 (1996), 247–254.
- Harter, D.E., Krishnan, M.S., Slaughter, S.A. Effects of process maturity on quality, cycle time, and effort in software product development. *Manage. Sci.* 46 4 (2000), 451–466.
- Hindley, R. The principal type-scheme of an object in combinatory logic. *Trans. Am. Math. Soc.* (1969), 29–60.
- Jump, M., McKinley, K.S. Cork: Dynamic memory leak detection for garbage-collected languages. In *ACM SIGPLAN Notices*, Volume 42 (2007). ACM, 31–38.
- Kleinschmager, S., Hanenberg, S., Robbes, R., Tanter, É., Stefik, A. Do static type systems improve the maintainability of software systems? An empirical study. In *2012 IEEE 20th International Conference on Program Comprehension (ICPC)* (2012). IEEE, 153–162.
- Li, Z., Tan, L., Wang, X., Lu, S., Zhou, Y., Zhai, C. Have things changed now? An empirical study of bug characteristics in modern open source software. In *ASID'06: Proceedings of the 1st Workshop on Architectural and System Support for Improving Software Dependability* (October 2006).
- Marques De Sá, J.P. *Applied Statistics Using SPSS, Statistica and Matlab*, 2003.
- Mayer, C., Hanenberg, S., Robbes, R., Tanter, É., Stefik, A. An empirical study of the influence of static type systems on the usability of undocumented software. In *ACM SIGPLAN Notices*, Volume 47 (2012). ACM, 683–702.
- Meyerovich, L.A., Rabkin, A.S. Empirical analysis of programming language adoption. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications* (2013). ACM, 1–18.
- Milner, R. A theory of type polymorphism in programming. *J. Comput. Syst. Sci.* 17, 3 (1978), 348–375.
- Mockus, A., Votta, L.G. Identifying reasons for software changes using historic databases. In *ICSM'00. Proceedings of the International Conference on Software Maintenance* (2000). IEEE Computer Society, 120.
- Odersky, M., Spoon, L., Venners, B. *Programming in Scala*. Artima Inc, 2008.
- Pankratius, V., Schmidt, F., Garretón, G. Combining functional and imperative programming for multicore software: An empirical study evaluating scala and java. In *Proceedings of the 2012 International Conference on Software Engineering* (2012). IEEE Press, 123–133.
- Petricek, T., Skeet, J. *Real World Functional Programming: With Examples in F# and C#*. Manning Publications Co., 2009.
- Pierce, B.C. *Types and Programming Languages*. MIT Press, 2002.
- Posnett, D., Bird, C., Devanbu, P. An empirical study on the influence of pattern roles on change-proneness. *Emp. Softw. Eng.* 16, 3 (2011), 396–423.
- Tan, L., Liu, C., Li, Z., Wang, X., Zhou, Y., Zhai, C. Bug characteristics in open source software. *Emp. Softw. Eng.* (2013).

Baishakhi Ray (rayb@virginia.edu),
Department of Computer Science,
University of Virginia, Charlottesville, VA.

**Daryl Posnett, Premkumar Devanbu,
and Vladimir Filkov** ([dpposnett@,
devanbu@cs., filkov@cs.]ucdavis.edu),
Department of Computer Science,
University of California, Davis, CA.

Copyright held by owners/authors.
Publication rights licensed to ACM.



The most useful and practical knowledge for building a variety of text data applications.

CS STUDENTS (Undergrad & Graduate)
LIBRARY & INFORMATION SCIENTISTS
TEXT DATA PRACTITIONERS

ChengXiang Zhai & Sean Massung (Authors)
University of Illinois at Urbana-Champaign

Text Data Management and Analysis covers the major concepts, techniques, and ideas in **information retrieval** and **text data mining**. It focuses on the practical viewpoint and **includes many hands-on exercises designed with a companion software toolkit** (i.e., MeTA) to help readers learn how to apply techniques of information retrieval and text mining to real-world text data. It also shows readers how to **experiment** with and **improve** some of the algorithms for interesting application tasks. The book can be used as a **text** for computer science undergraduates and graduates, library and information scientists, or as a **reference** for practitioners working on relevant problems in **managing and analyzing text data**.



<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/text>

Blog Ubiquity

INFORMATION EVERYWHERE



The newest ACM forum.

Contributions
that cover the vast
information-rich world
where computing
is embedded
everywhere.

ACM's *Ubiquity* is
the online magazine
oriented toward the
future of computing
and the people who
are creating it.

We invite you to
participate: leave
comments, vote for
your favorites, or
submit your own
contributions.

Captivating topics.

*Net Neutrality
and the Regulated
Internet*

*The End of Life
As We Know It*

*A Shortage of
Technicians*

*The Fractal
Software
Hypothesis*

*Your Grandfather's
Oldsmobile—NOT!*

*Superscalar
Smart Cities*



Association for
Computing Machinery

Visit us at
<http://ubiquity.acm.org/blog/>

California Institute of Technology Lecturer in Computing and Mathematical Sciences

The Department of Computing and Mathematical Sciences (CMS) at the California Institute of Technology invites applications for the position of Lecturer in Computing and Mathematical Sciences. This is a (non-tenure-track) career teaching position, with full-time teaching responsibilities. We seek to fill this position this coming academic year and the initial term of appointment can be up to three years.

The lecturer will teach introductory computer science courses including data structures, algorithms and software engineering, and will work closely with the CMS faculty on instructional matters. The ability to teach intermediate-level undergraduate courses in areas such as software engineering, computing systems and/or compilers is desired. The lecturer may also assist in other aspects of the undergraduate program, including curriculum development, academic advising, and monitoring research projects. The lecturer must have a track record of excellence in teaching computer science to undergraduates. In addition, the lecturer will have opportunities to participate in research projects in the department. An advanced degree in Computer Science or related field is desired but not required.

Applications will be accepted on an ongoing basis until the position is filled. Please view the application instructions and apply on-line at <https://applications.caltech.edu/job/cmslect>

The California Institute of Technology is an Equal Opportunity/Affirmative Action Employer. Women, minorities, veterans, and disabled persons are encouraged to apply.

Calvin College Tenure Track CS Faculty Position

The Department of Computer Science at Calvin College invites applications for a tenure track faculty position to begin August 2018, pending administrative approval. Our department features supportive colleagues, excellent facilities, (mostly) hardworking students, a dynamic colloquium series, and strong undergraduate programs in computer science, data science, digital communication, and information systems, including a BCS major accredited by ABET (abet.org). Applicants should have a PhD (or be near completion) in computer science or a related area, or have a master's degree and 5 years of related experience. We are especially interested in expanding our expertise in the areas of 3D modeling & animation, cybersecurity, data analytics & visualization, or machine learning, but individuals from all computing-related areas are encouraged to apply.

Calvin is a Christian comprehensive liberal arts college located in Grand Rapids, Michigan; it is one

of the largest Christian colleges in North America, and was named the #1 regional college in the Midwest for 2017 by U.S. News & World Report.

For more information and application information, see: https://cs.calvin.edu/documents/Tenure_Track_Faculty_Position.

Furman University Open-Rank Tenure Track Professor in Computer Science

The Department of Computer Science at Furman University invites applications for an open-rank tenure track position to begin in the fall of 2018. Candidates must have a Ph.D. in Computer Science or a closely related field, and all areas of specialty will be considered. The position requires teaching excellence, scholarly and professional activity involving undergraduates, effective institutional service, and a willingness to work with colleagues across disciplines.

The Department of Computer Science confers degrees with majors in Computer Science (B.S.) and Information Technology (B.S. and B.A.), an innovative, interdisciplinary program of study. The Department values teaching and research projects that bridge Computer Science with other disciplines, providing students with learning opportunities outside the classroom and in the community, and contributing to Furman's university-wide First Year Writing Seminar program. Furman Computer Science professors mentor undergraduates both formally and informally, and work to build a welcoming student-faculty community.

Furman University is a selective private liberal arts and sciences college committed to helping students develop intellectually, personally, and interpersonally and providing the practical skills necessary to succeed in a rapidly-changing world. Our recently-launched strategic vision, The Furman Advantage, promises students an individualized four-year pathway facilitated by team of mentors and infused with a rich and varied set of high-impact experiences outside the classroom that include undergraduate research, study away, internships, community-focused learning, and opportunities to engage across disciplines.

Furman is an Equal Opportunity Employer committed to increasing the diversity of its faculty and staff. The University aspires to create a community of people representing a multiplicity of identities including gender, race, religion, spiritual belief, sexual orientation, geographic origin, socioeconomic background, ideology, world view, and varied abilities. Domestic partners of employees are eligible for comprehensive benefits.

Applicants should submit a curriculum vitae, cover letter, statement of teaching philosophy and experience, statement of research interests,

an official copy of most recent transcripts, and a diversity statement that describes how your teaching, scholarship, mentoring and/or service might contribute to a liberal arts college community that includes a commitment to diversity as one of its core values. Three letters of recommendation should be sent separately. Review of applications will continue until the position is filled. To submit an application and letters of recommendation, please visit <http://jobs.furman.edu>.

Johns Hopkins University Full-Time Teaching Position

The Department of Computer Science at Johns Hopkins University seeks applicants for a full-time teaching position. This is a career-oriented, renewable appointment that is responsible for the development and delivery of courses primarily to undergraduate students both within and outside the major. Teaching faculty are also encouraged to engage in departmental and university service and may have advising responsibilities. Opportunities to teach graduate level courses may also be available, depending on the candidate's background. Extensive grading support is given to all instructors. The university has instituted a non-tenure track career path for full-time teaching faculty culminating in the rank of Teaching Professor.

Johns Hopkins is a private university known for its commitment to academic excellence and research. The Computer Science department is one of nine academic departments in the Whiting School of Engineering. We are located in Baltimore, MD in close proximity to Washington, DC and Philadelphia, PA. See the department webpages at <http://www.cs.jhu.edu> for additional information about the department, including undergraduate programs and current course descriptions.

Applicants for the position must have a Ph.D. in Computer Science or a closely related field, demonstrated excellence in and commitment to teaching, and excellent communication skills. Applicants should apply online at <https://academicjobsonline.org/ajo/jobs/9571>. Applications will be evaluated on a rolling basis. Questions should be directed to lecsearch2017@cs.jhu.edu.

The Johns Hopkins University is committed to active recruitment of a diverse faculty and student body. The University is an Affirmative Action/Equal Opportunity Employer of women, minorities, protected veterans and individuals with disabilities and encourages applications from these and other protected group members. Consistent with the University's goals of achieving excellence in all areas, we will assess the comprehensive qualifications of each applicant.

The Whiting School of Engineering and the Department of Computer Science are committed to building a diverse educational environment.

Johns Hopkins University Tenure-Track Faculty Positions

The Johns Hopkins University's Department of Computer Science seeks applicants for tenure-track faculty positions at all levels and across all areas of computer science. Particular emphasis is at the junior level and in the areas of systems, distributed systems, networks, system security, data center scale computing, and big data infrastructure. However, qualified applicants in all areas of computer science will be considered.

The Department of Computer Science has 30 full-time tenured and tenure-track faculty members, 8 research and 3 teaching faculty members, 150 PhD students, 225 MSE/MSSI students, and over 300 undergraduate students. There are several affiliated research centers and institutes including the Laboratory for Computational Sensing and Robotics (LCSR), the Center for Language and Speech Processing (CLSP), the JHU Information Security Institute (JHUISI), the Institute for Data Intensive Engineering and Science (IDIES), the Malone Center for Engineering in Healthcare (MCEH), and other labs and research groups. More information about the Department of Computer Science can be found at www.cs.jhu.edu and about the Whiting School of Engineering at <https://engineering.jhu.edu>. Qualifications and required materials can be found at <http://www.cs.jhu.edu/about/employment-opportunities/>.

Applicants should submit a curriculum vitae, a research statement, a teaching statement, three recent publications, and complete contact infor-

mation for at least three references. Applications must be made on-line at <https://academicjobsonline.org/ajob/jobs/9559>. Review of applications will begin in December 2017. While candidates who complete their applications by December 15, 2017 will receive full consideration, the department will consider exceptional applicants at any time. Questions should be directed to fsearch2017@cs.jhu.edu.

The Johns Hopkins University is committed to active recruitment of a diverse faculty and student body. The University is an Affirmative Action/Equal Opportunity Employer of women, minorities, protected veterans and individuals with disabilities and encourages applications from these and other protected group members. Consistent with the University's goals of achieving excellence in all areas, we will assess the comprehensive qualifications of each applicant.

The Whiting School of Engineering and the Department of Computer Science are committed to building a diverse educational environment.

Louisiana State University in Shreveport

Assistant or Associate Professor of Computer Science

The Department of Computer Science at Louisiana State University in Shreveport invites applications for a tenure-track Assistant/Associate Professor position. The successful candidate will teach 3 courses per semester (9 credit hours) beginning in the fall of 2018. Candidates must have

a Ph.D. in computer science or a closely related discipline, teaching experience, and strong research record or research potential.

Strong candidates in all areas of Computer Science are encouraged to apply but we are especially interested in cyber security and networking, mobile development, database, or big data analytics.

Applications should include a cover letter, curriculum vitae, research statement, a teaching statement, and a list of 3 references. Please send electronic copies of these documents to Alfred McKinney Alfred.McKinney@lsus.edu. Review of applications will begin on September 30th, 2017, and continue until the position is filled.

Please, visit <http://www.lsus.edu> for more details about LSUS and <http://www.lsus.edu/hr> for more information on this position and other employment opportunities.

LSUS is committed to achieving excellence through cultural diversity. The University actively encourages applications and/or nominations of women, persons of color, veterans and persons with disabilities. LSUS is an Affirmative Action, Equal Opportunity Employer.

Louisiana State University in Shreveport Chair of Computer Science

The Department of Computer Science at Louisiana State University in Shreveport invites applications for the position of Chair of the Department of Computer Science in the College of Arts and Sciences to begin in the fall of 2018.

Department of Electrical and Computer Engineering Graduate School of Engineering and Management Air Force Institute of Technology (AFIT) Dayton, Ohio



Department Head and Faculty Position

The Air Force Institute of Technology is seeking applications for both the position of Head, Department of Electrical and Computer Engineering (non-tenure-track), as well as a faculty position in Computer Science or Computer Engineering (tenure-track).

Applicants for the Department Head position must have an earned doctorate in electrical engineering, computer engineering, or computer science with credentials commensurate with the appointment as a full professor. A strong record of academic leadership experience is highly desirable. This is a non-tenure-track, 3 year renewable term position.

Applicants for the tenure-track faculty position must have an earned doctorate in computer science, computer engineering, electrical engineering, or closely related field. All areas and ranks will be considered. We are particularly interested in a cyber-security focus involving embedded and/or cyber physical systems security. The position requires teaching at the graduate level as well as establishing and sustaining a strong research program.

The Air Force Institute of Technology (AFIT) is the premier Department of Defense (DoD) institution for graduate education in science, technology, engineering, and management. The Department of Electrical and Computer Engineering offers accredited MS and PhD programs in Electrical Engineering, Computer Engineering, and Computer Science as well as an MS program in Cyber Operations.

Applicants for either position must be U.S. citizens. Full details on these positions, the department, applicant qualifications, and application procedures can be found at <https://www.afit.edu/ENG/>. Review of applications will begin on December 1, 2017. The United States Air Force is an equal opportunity, affirmative action employer.



上海科技大学
ShanghaiTech University



TENURE-TRACK AND TENURED POSITIONS

ShanghaiTech University invites highly qualified candidates to fill multiple tenure-track/tenured faculty positions as its core founding team in the School of Information Science and Technology (SIST). We seek candidates with exceptional academic records or demonstrated strong potentials in all cutting-edge research areas of information science and technology. They must be fluent in English. English-based overseas academic training or background is highly desired.

ShanghaiTech is founded as a world-class research university for training future generations of scientists, entrepreneurs, and technical leaders. Boasting a new modern campus in Zhangjiang Hightech Park of cosmopolitan Shanghai, ShanghaiTech shall trail-blaze a new education system in China. Besides establishing and maintaining a world-class research profile, faculty candidates are also expected to contribute substantially to both graduate and undergraduate educations.

Academic Disciplines: Candidates in all areas of information science and technology shall be considered. Our recruitment focus includes, but is not limited to: computer architecture, software engineering, database, computer security, VLSI, solid state and nano electronics, RF electronics, information and signal processing, networking, security, computational foundations, big data analytics, data mining, visualization, computer vision, bio-inspired computing systems, power electronics, power systems, machine and motor drive, power management IC as well as inter-disciplinary areas involving information science and technology.

Compensation and Benefits: Salary and startup funds are highly competitive, commensurate with experience and academic accomplishment. We also offer a comprehensive benefit package to employees and eligible dependents, including on-campus housing. All regular ShanghaiTech faculty members will join its new tenure-track system in accordance with international practice for progress evaluation and promotion.

Qualifications:

- Strong research productivity and demonstrated potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, Statistics, Applied Math, or related field);
- A minimum relevant (including PhD) research experience of 4 years.

Applications: Submit (in English, PDF version) a cover letter, a 2-page research plan, a CV plus copies of 3 most significant publications, and names of three referees to: sist@shanghaitech.edu.cn. For more information, visit <http://sist.shanghaitech.edu.cn/NewsDetail.asp?id=373>

Deadline: The positions will be open until they are filled by appropriate candidates.



CAL STATE
EAST BAY

CALIFORNIA STATE UNIVERSITY-EAST BAY

COLLEGE OF SCIENCE: COMPUTER SCIENCE

FACULTY EMPLOYMENT OPPORTUNITY

ASSISTANT PROFESSOR OF COMPUTER SCIENCE (GENERAL/CORE), TENURE-TRACK FACULTY

18-19 CS-GENERAL/CORE-TT [POSITION PS #00001011]

(2 positions)

THE UNIVERSITY: California State University, East Bay is known for award-winning programs, expert instruction, a diverse student body, and a choice of more than 100 career-focused fields of study. The ten major buildings of the Hayward Hills campus, on 342 acres, contain over 150 classrooms and teaching laboratories, over 177 specialized instructional rooms, numerous computer labs and a library, which contains a collection of over one million items. The University also has campuses in Concord and Oakland, as well as Online. With an enrollment of over 15,000 students and approximately 900 faculty, CSUEB is organized into four colleges: Letters, Arts, and Social Sciences; Business and Economics; Education and Allied Studies; and Science. The University offers bachelor's degrees in 50 fields, minors in 61 fields, master's degrees in 37 fields, 9 credentials programs, 12 certificate options, and 1 doctoral degree program. <http://www20.csueastbay.edu/> All California State University campuses, including CSUEB, will become smoke and tobacco-free effective September 1, 2017.

THE DEPARTMENT: The Department of Computer Science has 12 full-time faculty members, with a wide range of backgrounds and interests. The faculty is committed to teaching its undergraduate and Master's level students. In a typical quarter, the Department will offer over 30 undergraduate and about 20 graduate classes. Classes are offered both in day and evening. Classes are generally small, with many opportunities for faculty-student contact. The Department offers a variety of degrees: B.S. in Computer Science (with possible options in Networking and Data Communications, Software Engineering, or Computer Engineering), and M.S. in both Computer Science and Computer Networks. Currently, there are more than 500 undergraduate majors and over 250 students in the M.S. programs.

DUTIES OF THE POSITION (2 positions currently available): Teaching courses at B.S. and M.S. levels, curriculum development at both levels, and sustaining a research program. Please note that teaching assignments at California State University, East Bay include courses at the Hayward, Concord and Online campuses. In addition to teaching, all faculty have advising responsibilities, assist the department with administrative and/or committee work, and are expected to assume campus-wide committee responsibilities. The ideal candidate for this position is able to:

1. Teach a wide range of computer science courses including most or all of the core subject matter at both the undergraduate and graduate level.
2. Support offerings for undergraduate C.S. students including teaching courses, developing the undergraduate curriculum, and engaging undergraduate students in research.
3. Support offerings for graduate C.S. students including teaching courses, guiding M.S. theses, developing the graduate comprehensive examination, etc.
4. Advise Computer Science students.
5. Participate in departmental activities such as curriculum development, assessment, outreach, etc.
6. Develop and continue ongoing research activities, service and leadership.

RANK AND SALARY: Assistant Professor. Salary is dependent upon educational preparation and experience. Subject to budgetary authorization.

DATE OF APPOINTMENT: Fall Semester, 2018.

QUALIFICATIONS: Applicants must have a Ph.D. in Computer Science by Fall Semester 2018. Applicants must be able to teach undergraduate and master's level courses in most of the standard computer science core subjects. Candidates should demonstrate experience in teaching, mentoring, research, or community service that has prepared them to contribute to our commitment to diversity and excellence. Additionally, applicants must demonstrate a record of scholarly activity. Candidate's accomplishments should be commensurate with their professional level.

This University is fully committed to the rights of students, staff and faculty with disabilities in accordance with applicable state and federal laws. For more information about the University's program supporting the rights of our students with disabilities see: <http://www20.csueastbay.edu/af/departments/as/>

APPLICATION DEADLINE: Review of applications will begin November 1, 2017. The position, however, will be considered open until filled. Please submit a letter of application, which addresses the qualifications noted in the position announcement; a complete and current vita; names and email addresses of three references, three letters of recommendation, a statement of teaching philosophy, and evidence of teaching and research abilities via <http://apply.interfolio.com/42009>. See instructions on how to apply below. Applicants are strongly encouraged to also submit a one-page diversity statement that addresses how you engage a diverse student population in your teaching, research, mentoring, and advising.

Note: California State University, East Bay hires only individuals lawfully authorized to work in the United States. All offers of employment are contingent upon presentation of documents demonstrating the appointee's identity and eligibility to work in accordance with provisions of the Immigration Reform and Control Act. A background check (including a criminal records check and prior employment verification) must be completed and cleared prior to the start of employment.

For instructions on how to apply, please visit:
<https://help.interfolio.com/hc/en-us/articles/203701176-Job-Applicant-s-Guide-to-ByCommittee-Faculty-Search>

As an Equal Opportunity Employer, CSUEB does not discriminate on the basis of any protected categories: age, ancestry, citizenship, color, disability, gender, immigration status, marital status, national origin, race, religion, sexual orientation, or veteran's status. The University is committed to the principles of diversity in employment and to creating a stimulating learning environment for its diverse student body.

The Chair will teach 2 classes per semester (6 credit hours) and oversee Departmental operations, personnel, and resources. Additionally, the Chair will advance the Departmental vision of academic leadership and excellence and will represent the Department to academia, industry, and government for its continued vitality. The Chair will actively work with faculty in the Department and across the University to identify and pursue innovations in research, education, and service. Moreover, the Chair will be actively involved in strategic planning that would expand the influence of the Department in the broader academic community both within and outside the university, and will promote diversity within the Department and the institution.

The successful candidate must have earned a Ph.D. in Computer Science or a closely related field. The candidate should have a track record of research, classroom teaching, student mentorship, and funding from diverse sources. The candidate must provide evidence of scientific and organizational leadership, educational innovation, and have outstanding communication, interpersonal, and administrative skills.

Interested individuals should submit an application, including a cover letter summarizing qualifications and leadership approach, a vision statement for the Department, a research and student mentoring statement, a teaching statement, detailed curriculum vitae, and three letters of reference. Please send electronic copies of these documents to Dr. Alfred McKinney at Alfred.McKinney@lsus.edu.

Applications will be reviewed on a continuing basis starting on October 30th, 2017, until the position is filled and the posting removed from the LSUS HR Web site. We welcome questions related to this posting.

Please, visit <http://www.lsus.edu> for more details about LSUS and <http://www.lsus.edu/hr> for more information on this position and other employment opportunities.

LSUS is committed to achieving excellence through cultural diversity. The University actively encourages applications and/or nominations of women, persons of color, veterans and persons with disabilities. LSUS is an Affirmative Action, Equal Opportunity Employer.

Macalester College

Two Tenure-Track Assistant Professors of Computer Science

Macalester invites applications for two tenure-track positions at the assistant professor level to begin Fall 2018. Candidates who have, or are completing, a Ph.D. in Computer Science are preferred, but closely related fields may also be considered. We are especially interested in candidates who have a strong commitment to both teaching and research in an undergraduate liberal arts environment. This person will contribute to the teaching of our introductory, core and advanced courses, and mentor undergraduate research.

Macalester offers majors in Computer Science, Mathematics, and Applied Mathematics and Statistics, and minors in Computer Science,

Mathematics, and Statistics, as well as a new minor in Data Science. Typical class sizes range from 15 to 32 students. We encourage innovative pedagogy and curriculum and emphasize computer science's interdisciplinary connections. We have close relationships with several disciplines both within and beyond the sciences, and we are interested in candidates whose work spans disciplinary boundaries. Areas of highest priority include computer and data security and privacy, mobile and ubiquitous computing, computer networks and systems. For more information about our programs, see: <http://macalester.edu/mscs>

About Macalester

Macalester College is a highly selective, private liberal arts college in the vibrant Minneapolis-Saint Paul metropolitan area. The Twin Cities have a population of approximately three million, a rich arts community, strong local industries, an award-winning parks system, and are home to many colleges and universities, including the University of Minnesota. Macalester's diverse student body comprises over 2000 undergraduates from 40 states and the District of Columbia and over 90 nations. The College maintains a longstanding commitment to academic excellence with a special emphasis on internationalism, multiculturalism, and service to society. We are especially interested in applicants dedicated to excellence in teaching and research/creative activity within a liberal arts college community. As an Equal Opportunity employer supportive of affirmative efforts to achieve diversity among its faculty, Macalester College strongly encourages applications from women and members of underrepresented minority groups.

Applying

To apply via Academic Jobs Online submit (1) curriculum vitae, (2) graduate transcripts, (3) three letters of recommendation (at least one of which discusses your potential as a teacher), (4) a cover letter that addresses why you are interested in Macalester, (5) a statement of teaching philosophy, and (6) a research statement. Please contact Shilad Sen at ssen@macalester.edu with any questions about the position. Evaluation of applications will begin October 15, 2017 and continue until the position is filled.

Apply now: <https://www.macalester.edu/academics/mscs/compscitenture-trackjob.html>

Massachusetts Institute of Technology Faculty Positions

The Massachusetts Institute of Technology (MIT) Department of Electrical Engineering and Computer Science (EECS) seeks candidates for faculty positions starting in September 1, 2018, or on a mutually agreed date thereafter. Appointment will be at the assistant or untenured associate professor level. In special cases, a senior faculty appointment may be possible. Faculty duties include teaching at the undergraduate and graduate levels, research, and supervision of student research. Candidates should hold a Ph.D. in electrical engineering and computer science or a related field by the start of employment. We will consider candidates with research and teaching



ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Urbana-Champaign, IL

Head and Professor Department of Computer Science

The University of Illinois at Urbana-Champaign seeks a highly accomplished scholar and strategic leader as Head of the Department of Computer Science (CS). The new Head will lead a department that has continuously advanced the forefront of computing research and innovation since its inception, and whose faculty and alumni pioneered the modern computing era. The new Head will be uniquely positioned to build on the considerable strength and illustrious history of the department to lead the next chapter of the digital revolution. More details about the department can be found at <http://cs.illinois.edu>.

For a complete overview of this position, please visit <http://cs.illinois.edu/about-us/faculty-positions>. To ensure full consideration, applications should be received by **November 10, 2017**, but applications will be accepted until the position is filled.

The University has retained Witt/Kieffer, a national executive search firm, to assist in this recruitment. Nominations and applications, including cover letter, curriculum vita, and the names/contact information for three references, should be submitted electronically to Witt/Kieffer consultants John K. Thornburgh and Brian Bloomfield at the email address IllinoisCS@wittkieffer.com. The consultants can be reached by telephone, care of Donna Janulis, at 630-575-6131.

The University of Illinois conducts criminal background checks on all job candidates upon acceptance of a contingent offer.

Illinois is an EEO Employer/Vet/Disabled - www.inclusiveillinois.illinois.edu

WITT / KIEFFER

interests in any area of electrical engineering and computer science.

Candidates must register with the EECS search website at <https://school-of-engineering-faculty-search.mit.edu/eeecs/>, and must submit application materials electronically to this website. Candidate applications should include a description of professional interests and goals in both teaching and research. Each application should include a curriculum vitae and the names and addresses of three or more individuals who will provide letters of recommendation. Letter writers should submit their letters directly to MIT, preferably on the website or by mailing to the address below. Complete applications should be received by December 1, 2017. Applications will be considered complete only when both the applicant materials and **at least three letters of recommendation are received.**

It is the responsibility of the candidate to arrange reference letters to be uploaded at <https://school-of-engineering-faculty-search.mit.edu/eeecs/> by December 1, 2017.

Send all materials not submitted on the website to:

Professor Asu Ozdaglar
Interim Department Head, Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Room 38-435
77 Massachusetts Avenue
Cambridge, MA 02139

M.I.T. is an equal opportunity/affirmative action employer.

National University of Singapore Tenure-Track Faculty Positions in All Areas of Computer Science

The Department of Computer Science at the National University of Singapore (NUS) invites applications for tenure-track faculty positions in all areas of computer science. The Department enjoys ample research funding, moderate teaching loads, excellent facilities, and extensive international collaborations. We have a full range of faculty covering all major research areas in computer science and a thriving PhD program that attracts the brightest students from the region and beyond. More information is available at www.comp.nus.edu.sg/careers

NUS is an equal opportunity employer that offers highly competitive salaries, and is situated in Singapore, an English-speaking cosmopolitan city that is a melting pot of many cultures, both the east and the west. Singapore offers high-quality education and healthcare at all levels, as well as very low tax rates.

The Department is looking for candidates for all levels of tenured and tenure-track positions in any area of computer science. Candidates for senior positions should have an established record of outstanding, recognized research achievements, and thought leadership in his/her chosen area of computer science.

Candidates for Assistant Professor positions should demonstrate excellent research potential, and a strong commitment to teaching. Truly outstanding Assistant Professor applicants will also be considered for the prestigious Sung Kah Kay

Assistant Professorship as well as the President Assistant Professor.

Application Details:

- Submit the following documents (in a single PDF) online via: <https://faces.comp.nus.edu.sg>
 - A cover letter that indicates the position applied for and the main research interests
 - Curriculum Vitae
 - A teaching statement
 - A research statement
- Provide the contact information of 3 referees when submitting your online application, or, arrange for at least 3 references to be sent directly to csrec@comp.nus.edu.sg
- Application reviews will commence immediately and continue until positions are filled
- Please submit your application by 31 January 2018.

If you have further enquiries, please contact the Search Committee Chair, Weng-Fai Wong, at csrec@comp.nus.edu.sg

Santa Clara University Two Tenure-Track Assistant Professors in Computer Science

The Department of Mathematics and Computer Science at Santa Clara University invites applications for two tenure-track assistant professor positions in computer science. Our highest preference is in candidates with research interests in an area related to cybersecurity for the first position and



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to acmm mediasales@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

**Deadlines: 20th of the month/2 months prior to issue date.
For latest deadline info, please contact:
acmm mediasales@acm.org**

**Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at:
<http://jobs.acm.org>**

**Ads are listed for a period of 30 days.
For More Information Contact:**

**ACM Media Sales
at 212-626-0686 or
acmm mediasales@acm.org**



**FAKULTÄT
FÜR INFORMATIK**
Faculty of Informatics

Announcement of an open position at the Faculty of Informatics,
TU Wien, Austria

FULL PROFESSORSHIP of COMPUTER AIDED VERIFICATION (Successor of Helmut Veith)

The TU Wien (Vienna University of Technology) invites applications for a full professorship at the Faculty of Informatics.

The applicant is required to have an outstanding academic record in the field of Computer Aided Verification (CAV). Correctness, safety, and reliability of electronic systems are paramount in today's software-controlled world. The focus of the professorship on CAV will be on automated techniques to verify soft- and hardware. Besides a proven ability in CAV core methods (Computational Logic, Theoretical Computer Science), the candidate will also have a strong interdisciplinary background, especially in relation to Embedded Information Systems, Software Verification, Synthesis or Distributed Algorithms. This position will strengthen the area of Logic and Computation as well as form a link to other research foci of the faculty.

We offer excellent working conditions in an attractive research environment in a city with an exceptional quality of life.

For a more detailed announcement and information on how to apply, please go to: <http://www.informatik.tuwien.ac.at/vacancies>

Application Deadline: October 16, 2017

an area related to algorithms for the second position. Strong candidates with research interests in artificial intelligence and software aspect of data science will be considered as well. The successful candidates will demonstrate not only potential for excellent undergraduate teaching, but also promise in sustained research with opportunities to involve undergraduates, mentoring or recruiting underrepresented groups in computer science, and service to the department, College or University. **Positions available starting in September 2018.** Ph.D. or equivalent required by September 2018.

The closing date for applications is December 1, 2017 at 3 pm Pacific time. Undergraduate teaching only.

Santa Clara University, located in California's Silicon Valley, is a comprehensive, Jesuit, Catholic university, and an AA/EEO employer.

For more information or to apply, visit <https://jobs.scu.edu/postings/6211>.

Stanford University, Graduate School of Business

Faculty Positions in Operations, Information and Technology

The Operations, Information and Technology (OIT) area at the Graduate School of Business, Stanford University, is seeking qualified applicants for full-time, tenure-track positions, starting September 1, 2018. All ranks and relevant disciplines will be considered. Applicants are considered in all areas of Operations, Information and Technology (OIT) that are broadly defined to

include the analytical and empirical study of technological systems, in which technology, people, and markets interact. It thus includes operations, information systems/technology, and management of technology. Applicants are expected to have rigorous training in management science, engineering, computer science, economics, and/or statistical modeling methodologies. Candidates with strong empirical training in economics, behavioral science or computer science are encouraged to apply. The appointed will be expected to do innovative research in the OIT field, to participate in the school's PhD program, and to teach both required and elective courses in the MBA program. Junior applicants should have or expect to complete a PhD by September 1, 2018.

Applicants should submit their applications electronically by visiting the web site <http://www.gsb.stanford.edu/recruiting> and uploading their curriculum vitae, research papers and publications, and teaching evaluations, if applicable, on that site. **For an application to be considered complete, all applicants must submit a CV, a job market paper and arrange for three letters of recommendation to be submitted by November 15, 2017.** For questions regarding the application process, please send an email to Faculty_Recruiter@gsb.stanford.edu.

Stanford is an equal employment opportunity and affirmative action employer. All qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, disability, protected veteran status, or any other characteristic protected by law.

Tufts University

Dept. of Computer Science

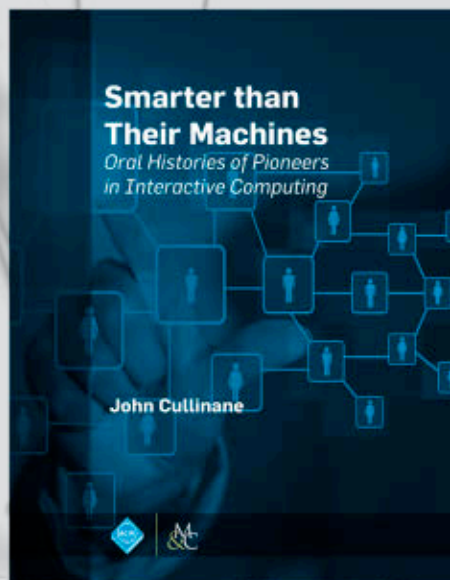
Tenure-Track Assistant Professor in Computer Science

The Department of Computer Science at Tufts University invites applications for multiple tenure-track faculty positions to begin in September 2018. We are looking for engaged and engaging researchers and teachers with a strong vision who can build and maintain a high-quality research program at Tufts and whose research will both connect with some of our current faculty and extend into new areas. We are seeking candidates at the rank of Assistant Professor but exceptional candidates at the rank of Associate or full Professor will also be considered.

We are especially interested in candidates with research in Artificial Intelligence/Machine Learning/Robotics, Security, and Systems for Data Science, where these areas are broadly construed. Exceptional candidates in other areas will be considered as well.

Please submit your application online through Interfolio at <https://apply.interfolio.com/43666>. You can contact help@interfolio.com with questions.

Review of applications will begin December 15, 2017 and will continue until the position is filled. For more information about the department or the position, please visit our web page at <http://engineering.tufts.edu/computer-science-positions>. Inquiries should be emailed to ttsearch@cs.tufts.edu.



A personal walk down the computer industry road. BY AN EYEWITNESS.

Smarter Than Their Machines: Oral Histories of the Pioneers of Interactive Computing is based on oral histories archived at the Charles Babbage Institute, University of Minnesota. These oral histories contain important messages for our leaders of today, at all levels, including that government, industry, and academia can accomplish great things when working together in an effective way.



ISBN: 978-1-62705-550-5 DOI: 110.1145/2663015

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/acm>

modern computing era. The new Head will be uniquely positioned to build on the considerable strength and illustrious history of the department to lead the next chapter of the digital revolution. More details about the department can be found at <http://cs.illinois.edu>.

The Head is responsible for visioning, strategic planning, operations, finance, academic affairs, external relations, and advancement, and is a tenured Professor in the department. The successful candidate will be committed to enhancing the University's education, research, and service missions and will possess the scholarly record, leadership skills, and strategic capacity to advance the department. Additional essential qualifications include successful administrative experience in a university, industry, or government environment, the ability to effectively engage a broad range of internal and external constituencies, and a commitment to diversity.

The Head position is a full-time, twelve-month administrative appointment accompanied by a full-time, tenured Professor appointment with full University benefits. The desired start date for this position is as soon as possible. Applicants may be interviewed before the closing date; however, no hiring decision will be made until after that date. To ensure full consideration, applications should be received by **November 10, 2017**, but applications will be accepted until the position is filled. Salary is negotiable and commensurate with skills and experience.

The University has retained Witt/Kieffer, a national executive search firm, to assist in this recruitment. Nominations and applications, including cover letter, curriculum vita, and the names/contact information for three references, should be submitted electronically to Witt/Kieffer consultants John K. Thornburgh and Brian Bloomfield at the email address IllinoisCS@wittkieffer.com. The consultants can be reached by telephone, care of Donna Janulis, at 630-575-6131.

The University of Illinois conducts criminal background checks on all job candidates upon acceptance of a contingent offer.

The University of Illinois is an Equal Opportunity, Affirmative Action employer. Minorities, women, veterans and individuals with disabilities are encouraged to apply. For more information, visit <http://go.illinois.edu/EEO>. To learn more about the University's commitment to diversity, please visit <http://www.inclusiveillinois.illinois.edu>.

University of Minnesota Two Tenure-Track Positions

The Department of Industrial and Systems Engineering at the University of Minnesota invites applications for two tenure-track faculty positions starting in Fall 2018.

Applicants at all ranks will be considered. We seek candidates with a strong methodological foundation in Operations Research, Industrial Engineering and a demonstrated interest in applications including, but not limited to: business analytics, energy and the environment, healthcare and medical applications, transportation and logistics, supply chain management, and service systems. Applicants must hold a Ph.D. in Industrial Engineering, Operations Research, Operations Management or a closely related discipline and have demonstrated the potential to conduct a vigorous and significant research program as evidenced by their publication

record and supporting letters from recognized leaders in the field. Senior applicants should have a particularly strong record of research and teaching accomplishments, scientific leadership and creativity.

The University of Minnesota is located in the heart of the vibrant Minneapolis-St. Paul metropolitan area, which is consistently rated as one of America's best places to live and is home to many leading companies. The Department of Industrial and Systems Engineering is within the College of Science and Engineering at the University of Minnesota.

Applicants are encouraged to apply by November 1, 2017. Review of applications will begin immediately and will continue until the position is filled. Additional information and application instructions can be found at http://www.isye.umn.edu/news/Search_f2017.shtml. The University of Minnesota is an equal opportunity educator and employer.

University of Nevada, Reno Assistant or Associate Professor

The Department of Computer Science and Engineering (CSE) at the University of Nevada, Reno (UNR) invites applications for a Tenure-track Faculty position starting July 1, 2018. The Department seeks highly qualified candidates in games, hardware, or in areas that extend or complement the Department's existing strengths or fulfill Department needs. The position is at the Assistant or Associate Professor level. The new hire will work with existing faculty to strengthen research, attract research funding, teach courses, and enhance our graduate and undergraduate programs.

The rapidly expanding and dynamic Department of Computer Science and Engineering has added nine positions over the last five years and expects to add more. Several faculty have NSF CAREER awards and play lead roles in multiple state-wide and national multi-million dollar NSF awards. In addition to federal support from DoD, DHS, DoE, and NASA, companies like Google, Microsoft, Ford, AT&T, Nokia, and Honda support our research.

In the last five years, the College of Engineering has witnessed unprecedented growth in student enrollment and number of faculty positions. The College is positioned to further enhance its growth of its students, faculty, staff, facilities as well as its research productivity and its graduate and undergraduate programs.

Thanks to this substantial growth in both student enrollment and tenure-track faculty positions, the College of Engineering has received funding to build a new engineering building, scheduled to be completed in 2020. The new engineering building provides both additional space critically needed by the College and the modern facilities capable of supporting advanced research and laboratory space. This building will allow the College to pursue its strategic vision, serve Nevada and the nation, and educate future generations of engineering professionals.

The University of Nevada, Reno recognizes that diversity promotes excellence in education and research. We are an inclusive and engaged community and recognize the added value that students, faculty, and staff from different backgrounds bring to the educational experience.

Interested candidates must apply online to www.unrsearch.com/postings/25237. Application process includes: a detailed letter of application, curriculum vitae, statement of teaching phi-

losophy, state of research and plans, and contact information for three professional references.

Review of applications will begin on November 1, 2017 and continue until the search closes on December 31, 2017. Inquiries should be directed to Ms. Lisa Cody, lacody@unr.edu.

EEO/AA Women and under-represented groups, individuals with disabilities, and veterans are encouraged to apply.

University of Oregon Dept. of Computer and Information Science Faculty Positions

The University of Oregon's Computer and Information Science (CIS) Department invites applications for two tenure-track faculty positions at the rank of Assistant Professor, to begin in September 2018. We seek candidates specializing in high-performance computing or data science. We are especially interested in scholars who will enhance/complement the department's existing strengths in these areas. Applicants whose research addresses security and/or privacy issues in these sub-disciplines are of particular interest.

CIS is a diverse and growing department with strengths in networking and distributed systems, data science, and high-performance computing. We offer a stimulating, friendly environment for collaborative research both within the department, which expects to grow substantially in the next few years, and with other units on campus; for example, the department plays a key role in the Knight Campus for Accelerating Scientific Impact. The department hosts two interdisciplinary research centers, the Center for Cyber Security and Privacy and the NeuroInformatics Center. Successful candidates have access to a new, state-of-the-art high-performance computing facility. CIS is part of the College of Arts and Sciences and is housed within the Lorry Lokey Science Complex. The department offers B.S., M.S. and Ph.D. degrees. More information about the department, its programs and faculty can be found at <https://cs.uoregon.edu/>.

Applicants must have a Ph.D. in computer science or closely related field, a demonstrated record of excellence in research, and a strong commitment to teaching. A successful candidate will be expected to conduct a vigorous research program and to teach at both the undergraduate and graduate levels. Additionally, successful candidates will support and enhance a diverse learning and working environment. Salary is competitive.

Candidates are asked to apply online at <https://academicjobsonline.org/ajob/jobs/9499> by submitting a cover letter, a curriculum vitae, a research statement, a teaching statement, and the contact details for a minimum of three referees, by 15 December 2017, or until the post has been filled. If you are unable to use this online resource, please contact faculty.search@cs.uoregon.edu to arrange alternate means of application submission.

The University of Oregon is dedicated to the goal of building a culturally diverse and pluralistic faculty committed to teaching and working in a multicultural environment and strongly encourages applications from minorities, women, and people with disabilities. Applicants are requested to include in their cover letter information about how they will further this goal. In particular, candidates should describe previous activities mentoring minorities, women, or members of other underrepresented groups.

ACM LEARNING CENTER

RESOURCES
FOR LIFELONG LEARNING

learning.acm.org



Online Courses from Skillsoft

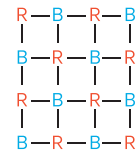
Online Books from Safari, Books24x7,
Morgan Kaufmann and Syngress

Webinars on today's
hottest topics in computing

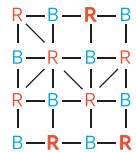


Association for
Computing Machinery

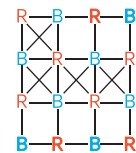
[CONTINUED FROM P. 112] goal of any solution is to minimize swaps.



Solution to challenge. Numbering the rows from top to bottom and left to right, we can build roads between red towns from (1,1) to (2,2), (2,4) to (3,3), (2,2) to (3,3), and (3,1) to (2,2). This leaves the red towns at (1,3), (4,2), and (4,4) surrounded by blue towns, highlighted in bold, in the following grid



Now build roads between the blue towns from (1,2) to (2,1), from (2,1) to (3,2), from (2,3) to (3,2), and from (2,3) to (3,4). This leaves certain towns isolated from their own colors, like those highlighted in bold in this grid



The two sides then need just three swaps: red and blue on top and the pairs on the bottom.

Upstart 1. Suppose you are given an arbitrary planar graph of connections and an arbitrary red/blue coloring of nodes. You are also given a budget of r planar edges you can add. Can you create an algorithm (and implementation) that will perform a minimum number of swaps to achieve partitioned peace? If so, please explain it in pseudo-code and send links to platform-independent software.

Upstart 2. Now consider the same question as in Upstart, 1 but allow non-planar edges.

All are invited to submit their solutions to upstartpuzzles@acm.acm.org; solutions and discussion will be posted at <http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html>

Dennis Shasha (dennisshasha@yahoo.com) is a professor of computer science in the Computer Science Department of the Courant Institute at New York University, New York, as well as the chronicler of his good friend the omniheurist Dr. Ecco.

Copyright held by the author.



DOI:10.1145/3133244

Dennis Shasha

Upstart Puzzles

Partitioned Peace

IN A MYTHICAL land of rhetorically encouraged antagonism, different factions manage to co-exist, though poorly. Imagine a set of red and blue hill towns connected by a network of roads. People in the red towns deal well with one another. People in the blue towns deal well with one another. But when a person from a red town travels through a blue town or vice versa, things can get unpleasant. The leaders of the red and the blue towns get together and decide the best way to resolve their differences is to perform a series of swaps in which the inhabitants of k red towns swap towns with the inhabitants of k blue towns with the end result that a person from a blue town can visit any other blue town without passing through a red town and likewise for a person from a red town. We call such a desirable state “partitioned peace.” The goal is to make k as small as possible.

Warm-Up 1. Given the configuration in the figure here, what is the minimum number of swaps needed to achieve partitioned peace?

Solution to Warm-Up 1. Two swaps are sufficient: Red_1 with Blue_7 and Red_3 with Blue_4.

Because exchanging town populations is painful for the people who must move, the leaders seek other arrangements; they are willing to build, for example, a certain number of roads to reduce the number of swaps.

Warm-Up 2. Given the configuration in the figure, what is the minimum number of swaps needed to achieve partitioned peace if you were able to build a single new road?

Solution to Warm-Up 2. Build a road between Blue_7 and Red_1 and then swap Red_1 with Blue_4. The red town dwellers can travel to other red towns



What is the minimum number of towns you can exchange so red and blue travelers never need to cross the other color's towns?

without crossing through blue towns. Likewise, the blue town dwellers can travel to other blue towns without hav-

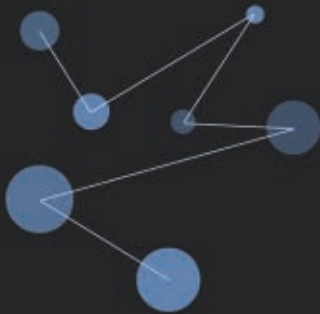
ing to cross through red towns.

Challenge: So far, we have considered only a very simple configuration of towns; now consider that the red and blue towns alternate like the squares of a four-by-four checkerboard. Every town is connected to all its vertical and horizontal neighbors. You may build eight new roads between diagonally neighboring towns. Where should the roads go? And after the roads are built, which towns should swap populations to minimize the number of swaps needed to achieve partitioned peace, where the swaps are between towns directly connected by roads? The [CONTINUED ON P. 111]

Can you create an algorithm that will perform a minimum number of swaps to achieve partitioned peace?

ETRA

ACM SYMPOSIUM ON EYE TRACKING
RESEARCH & APPLICATIONS



June 14-17, 2018
SWPS University of Social
Sciences and Humanities
Warsaw, Poland

The tenth anniversary of the ACM Symposium on Eye Tracking Research & Applications (ETRA 2018) will be held for the first time in Europe! This edition of ETRA will introduce several innovations such as co-located workshops, invited sessions and a special journal issue based on selected invited papers.

The ETRA conference series focuses on eye movement research and applications across a wide range of disciplines including computer science, human-computer interaction, visualization, biomedical research, virtual reality and psychology.

Join us in Warsaw to celebrate 20 years of eye tracking research at ETRA!

Important dates

Papers & Notes

Jan 12 2018 Paper Abstracts due
Jan 26 2018 Papers & Notes due
Mar 04 2018 Rebuttals due
Apr 13 2018 Camera ready papers due
Apr 23 2018 Final notifications to authors

Demo/Video & Doctoral Symposium

April 02 2018 Extended abstracts due
April 16 2018 Camera ready from technical abstracts due
April 23 2018 Final notifications to authors technical abstracts due

General Chairs: Bonita Sharif (Youngstown State University, USA) & Krzysztof Krejtz (SWPS University of Social Sciences and Humanities, Poland)



@ETRAConference



@ETRA_conference

WEBSITE: <http://etra.acm.org>





10th
Edition

CONFERENCE 27 – 30 November 2017
EXHIBITION 28 – 30 November 2017
BITEC, Bangkok, Thailand

THE CELEBRATION OF LIFE & TECHNOLOGY

The 10th ACM SIGGRAPH Conference and Exhibition
on Computer Graphics and Interactive Techniques in Asia



Register online by 15 October 2017,
& enjoy early bird discounts of up to

20%

👉 SA2017.SIGGRAPH.ORG/REGISTRATION

Sponsored by



Organized by

