

COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

03/2018 VOL.61 NO.03



A Programmable Programming Language

How Can We Trust a Robot?

Policy Intervention Fosters Cybersecurity Intervention

Here Comes Everyone ... to *Communications*

Q&A with Yann LeCun

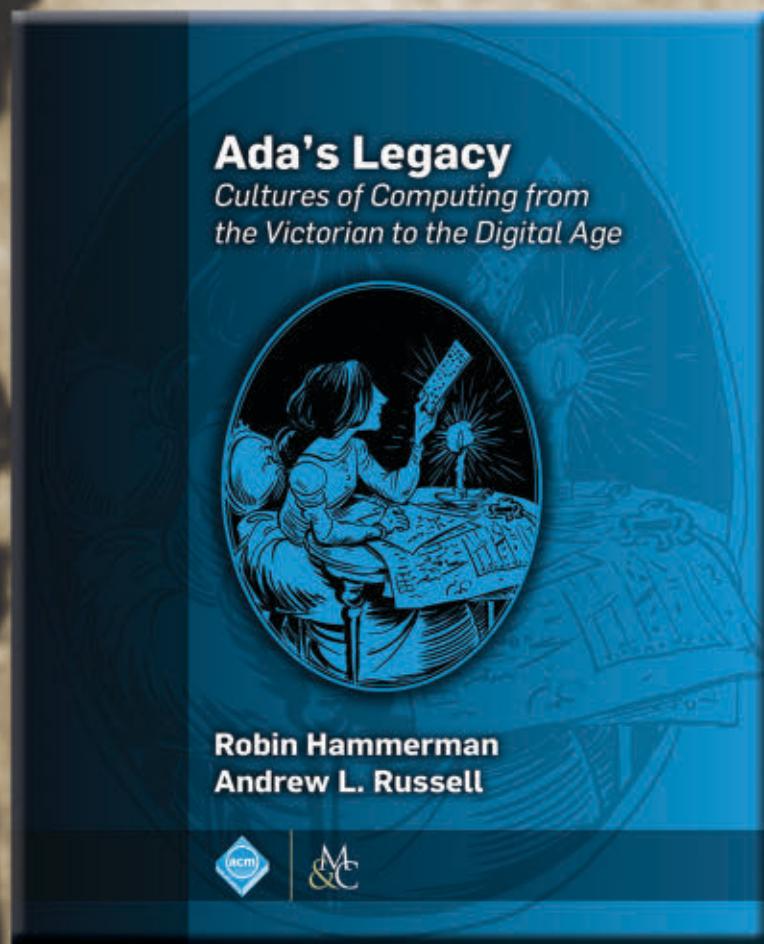
Association for Computing Machinery



Robin Hammerman and Andrew L. Russell

Ada's Legacy

Cultures of Computing from the
Victorian to the Digital Age



INSPIRING MINDS
FOR 200 YEARS

Ada's Legacy illustrates the depth and diversity of writers, things, and makers who have been inspired by Ada Lovelace, the English mathematician and writer.

The volume commemorates the bicentennial of Ada's birth in December 1815, celebrating her many achievements as well as the impact of her work which reverberated widely since the late 19th century. This is a unique contribution to a resurgence in Lovelace scholarship, thanks to the expanding influence of women in science, technology, engineering and mathematics.

ACM Books is a new series of high quality books for the computer science community, published by the Association for Computing Machinery with Morgan & Claypool Publishers.



ISBN: 978-1-97000-148-8 DOI: 10.1145/2809523
<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/acm>

Marquette University's Third Annual

ETHICS OF BIG DATA SYMPOSIUM

The emerging world of big data brings with it ethical, social and legal issues. Are you prepared to navigate the challenges and the opportunities?

Friday, April 27

8 a.m. – Noon

Marquette University

Milwaukee, Wisconsin

For event information and registration, go to marquette.edu/ethics-of-big-data
or contact Dr. Thomas Kaczmarek at thomas.kaczmarek@marquette.edu or 414.288.6734.

The deployment of big data brings desirable opportunities to understand, recommend and advise. But the sensitivity of personal data and unintended consequences of algorithmic decisions present us with ethical and moral decisions. The symposium will cover ethical and legal considerations for practitioners, including discussions and dilemmas of agency, fairness, public perception and privacy.

Hosted by Northwestern Mutual.



**MARQUETTE
UNIVERSITY**

BE THE DIFFERENCE.

COMMUNICATIONS OF THE ACM

Departments

- 5 **Editor's Letter**
Here Comes Everybody ... to Communications
By Andrew A. Chien

- 7 **Cerf's Up**
Unintended Consequences
By Vinton G. Cerf

- 9 **Vardi's Insights**
A Declaration of the Dependence of Cyberspace
By Moshe Y. Vardi

- 10 **Letters to the Editor**
Keep the ACM Code of Ethics As It Is

- 12 **BLOG@CACM**
The Costs and Pleasures of a Computer Science Teacher
Mark Guzdial considers the enormous opportunity costs of computer science teachers, while Bertrand Meyer ponders the pleasures of arguing with graduate students.

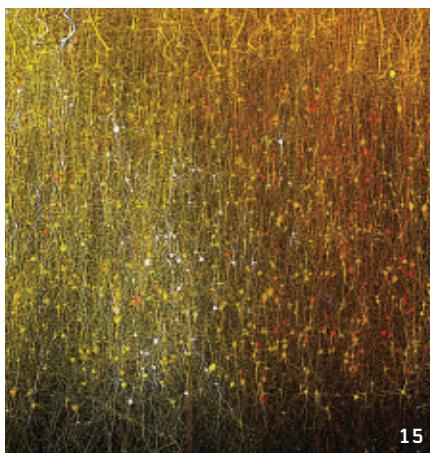
- 43 **Calendar**

- 117 **Careers**

Last Byte

- 120 **Q&A**
The Network Effect
The developer of convolutional neural networks looks at their impact, today and in the long run.
By Leah Hoffmann

News



15

- 15 **In Pursuit of Virtual Life**
Scientists are simulating biological organisms and replicating evolution in the lab. How far can they expand the boundaries of virtual life?
By Samuel Greengard

- 18 **The Construction Industry in the 21st Century**
Three-dimensional printing and other new technologies are revitalizing the business of building buildings.
By Keith Kirkpatrick

- 21 **The State of Fakery**
How digital media could be authenticated, from computational, legal, and ethical points of view.
By Esther Shein

Viewpoints

- 24 **Privacy and Security**
Making Security Sustainable
Can there be an Internet of durable goods?
By Ross Anderson

Viewpoints, cont'd.

- 27 **Legally Speaking**
Will the Supreme Court Nix Reviews of Bad Patents?
Considering the longer-term implications of a soon-to-be-decided U.S. Supreme Court case.
By Pamela Samuelson

- 30 **Computing Ethics**
Ethics Omission Increases Gases Emission
A look in the rearview mirror at Volkswagen software engineering.
By Simon Rogerson

- 33 **The Profession of IT**
The Computing Profession
Taking stock of progress toward a computing profession since this column started in 2001.
By Peter J. Denning

- 36 **Viewpoint**
Impediments with Policy
Interventions to Foster Cybersecurity
A call for discussion of governmental investment and intervention in support of cybersecurity.
By Fred B. Schneider

- 39 **Viewpoint**
Responsible Research with Crowds: Pay Crowdworkers at Least Minimum Wage
High-level guidelines for the treatment of crowdworkers.
By M. Six Silberman, Bill Tomlinson, Rochelle LaPlante, Joel Ross, Lilly Irani, and Andrew Zaldivar

- 42 **Viewpoint**
Computational Social Science ≠ Computer Science + Social Data
The important intersection of computer science and social science.
By Hanna Wallach

Practice

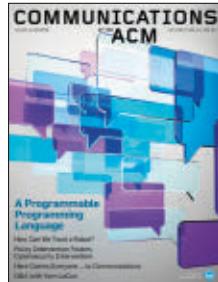
46

- 46 Bitcoin's Underlying Incentives**
The unseen economic forces that govern the Bitcoin protocol.
By Yonatan Sompolinsky and Aviv Zohar

- 54 Operational Excellence in April Fools' Pranks**
Being funny is serious work.
By Thomas A. Limoncelli

- 58 Monitoring in a DevOps World**
Perfect should never be the enemy of better.
By Theo Schlossnagel

 Articles' development led by **ACM Queue**
queue.acm.org



About the Cover:
This month's cover story explores the Racket language project, a programming language designed to support language-oriented programming. The story of this project, now celebrating 20 years, is told by its creators beginning on p. 62. Cover illustration by Chris Labrooy.

Contributed Articles

72

- 62 A Programmable Programming Language**
As the software industry enters the era of language-oriented programming, it needs programmable programming languages.
By Matthias Felleisen, Robert Bruce Findler, Matthew Flatt, Shriram Krishnamurthi, Eli Barzilay, Jay McCarthy, and Sam Tobin-Hochstadt



Watch the authors discuss their work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/a-programmable-programming-language>

- 72 The Wisdom of Older Technology (Non)Users**
Older adults consistently reject digital technology even when designed to be accessible and trustworthy.
By Bran Knowles and Vicki L. Hanson

- 78 Evolution Toward Soft(er) Products**
As software becomes a larger part of all products, traditional (hardware) manufacturers are becoming, in essence, software companies.
By Tony Gorscheck

Review Articles**86 How Can We Trust a Robot?**

If intelligent robots take on a larger role in our society, what basis will humans have for trusting them?
By Benjamin Kuipers



Watch the author discuss his work in this exclusive *Communications* video.
<https://cacm.acm.org/videos/how-can-we-trust-a-robot>

Research Highlights

- 98 Technical Perspective**
A Graph-Theoretic Framework for Traces Task Planning
By Nicole Immorlica

- 99 Time-Inconsistent Planning: A Computational Problem in Behavioral Economics**
By Jon Kleinberg and Sigal Oren

- 108 Technical Perspective**
On Heartbleed: A Hard Beginning Maketh a Good Endyng
By Kenny Paterson

- 109 Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed**
By Liang Zhang, David Choffnes, Tudor Dumitras, Dave Levin, Alan Mislove, Aaron Schulman, and Christo Wilson





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.

Acting Executive Director
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 Cappo
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
Chris Hankin
ACM India Council
Madhavan Mukund
ACM China Council
Yunhai Liu

PUBLICATIONS BOARD

Co-Chairs
Jack Davidson; Joseph Konstan
Board Members
Phoebe Ayers; Anne Condon; Nikil Dutt; Roch Guerrin; Chris Hankin; Yannis Ioannidis; XiangYang Li; Sue Moon; Michael L. Nelson; Sharon Oviatt; Eugene H. Spafford; Stephen N. Spencer; Alex Wade; Julie R. Williamson

ACM U.S. Public Policy Office

Adam Eisgrau,
Director of Global Policy and Public Affairs
1701 Pennsylvania Ave NW, Suite 300,
Washington, DC 20006 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Jake Baskin
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 Rosenblom

Senior Editor/News

Lawrence M. Fisher

Web Editor

David Roman

Rights and Permissions

Deborah Cotton

Editorial Assistant

Jade Morris

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquit

Production Manager

Bernadette Shadé

Advertising Sales Account Manager

Ilia Rodriguez

Columnists

David Anderson; Phillip G. Armour; Michael Cusumano; Peter J. Denning; Mark Guzdiel; 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

acmhelp@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

acmmediasales@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

Deputy to the Editor-in-Chief

Lihan Chen
cacm.deputy.to.eic@gmail.com

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

Stefan Bechtold; Michael L. Best; Judith Bishop; Andrew W. Cross; Mark Guzdiel; Richard Ladner; Carl Landwehr; Beng Chin Ooi; Francesca Rossi; 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; Kim Bruce; Alan Bundy; Peter Buneman; Carl Gutwin; Yannis Ioannidis; Gal A. Kaminka; Ashish Kapoor; Kristin Lauter; Igor Markov; Bernhard Nebel; Lionel M. Ni; Adrian Perrig; Marie-Christine Rousset; Krishnan Sabnani; m.c. schraefel; Ron Shamir; Alex Smola; Josep Torrellas; Sebastian Uchitel; Michael Vitale; Hannes Werthner; Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Shriram Krishnamurthi

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; Alan Halevy; Sven Koenig; Steve Marschner; Greg Morrisett; Tim Roughgarden; Guy Steele, Jr.; Margaret H. Wright; Nicholai Zeldovich; Andreas Zeller

WEB

Chair

James Landay

Board Members

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

ACM Copyright Notice

Copyright © 2018 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 acmhelp@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 USA.



Association for Computing Machinery





DOI:10.1145/3183638

Andrew A. Chien

Here Comes Everybody... to Communications

I am pleased to announce a new *Communications of the ACM* initiative with the ambitious goal of expanding the *Communications* community globally.

I hope it means “Here Comes Everybody to *Communications*.^a Why bring everybody to *Communications*? To include important voices and perspectives in the conversation about the present and future of computing. With the proliferation of computing into every industry, every product, and every aspect of society, not only has computing spread throughout the society and economy of every nation, but the computing profession has spread into communities around the globe.

One natural consequence is that invention and innovation in computing, once concentrated in a few regions, is now a global enterprise. And, while the technical foundations of computing may be universal,^b along with technical challenges of functionality, scale, reliability, and perhaps usability; increasingly, the design of many of a system’s most important aspects—how they relate to society, government, structure of commerce, and individual enlightenment and perspective as well as fundamental

choices about security, privacy, free speech, and control—reflect distinctive regional, national, and community culture.

Communications, the flagship publication of world’s leading computing professional society, should be an inclusive forum, spanning this community. It should be a universal forum, an inclusive, global community, with active participation from everyone.

To that end, I am pleased to announce a *Communications* global initiative. Its goal is to give deeper insight, focused coverage, and elevate distinctive and compelling highlights of computing drawn from regions around the world. This initiative will add a 20–30 page special section to a few issues of *Communications* each year. Each special section will be a collection of short pieces, focused on a region and chartered to represent the best of computing leadership and distinctive development. We will bring a sharp focus on:

- ▶ Leading technical and research advances and activities;
- ▶ Leading and emerging industry and research players;
- ▶ Innovation and shape of computing in the region; and,
- ▶ Unique challenges and opportunities ... and by doing so enrich the entire computing community’s perspective!

Communications’ global initiative will visit regions around the world in turn, shifting its spotlight to match the pace and impact of interesting develop-

ments in computing. We hope to revisit regions about once every two years.

I am pleased to report that we have already begun. The first special section will focus on China, where we have convened an extraordinary team of industry and academic leaders committed to attend the kick-off meeting (set for March 8th at the UChicago Beijing Center), and we are actively planning successors in other parts of the world.

How we do this is instrumental. The special sections will be led by a regional team who will nominate, select, and drive authorship of the section’s content. By design, this will encourage active participation of a growing global community in *Communications*. To drive creation of the regional teams and the entire series of special sections, we are adding a new section to the Editorial Board of *Communications*. Serendipitously, this creates new opportunities for you to volunteer and contribute to the magazine.

Expect to hear more about this late in 2018 when we print the first special section!

Andrew A. Chien, EDITOR-IN-CHIEF

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

Copyright held by author.

ACM Books. In-depth. Innovative. Insightful.

ACM and Morgan & Claypool Publishers present ACM Books: an all-new series of educational, research and reference works for the computing community. Inspired by the need for high-quality computer science publishing at the graduate, faculty and professional levels, ACM Books is affordable, current, and comprehensive in scope.

ACM Books collections are available under an ownership model with archival rights included. We invite you to learn more about this exciting new program.

For more info please visit
<http://books.acm.org>

or contact ACM at
ACMbooks-Info@acm.org

**Association for
Computing Machinery**
2 Penn Plaza, Suite 701
New York, NY 10121-0701, USA
Phone: +1-212-626-0658
Email: acmbooks-info@acm.org



**Morgan & Claypool
Publishers**
1210 Fifth Avenue, Suite 250
San Rafael, CA 94901, USA
Phone: +1-415-462-0004
Email: info@morganclaypool.com





DOI:10.1145/3184402

Vinton G. Cerf

Unintended Consequences

WHEN THE INTERNET was being developed, scientists and engineers in academic and research settings drove the process. In their world, information was a medium of exchange. Rather than buying information from each other, they exchanged it. Patents were not the first choice for making progress; rather, open sharing of designs and protocols were preferred. Of course, there were instances where hardware and even software received patent and licensing treatment, but the overwhelming trend was to keep protocols and standards open and free of licensing constraints. The information-sharing ethic contributed to the belief that driving down barriers to information and resource sharing was an important objective. Indeed, the Internet as we know it today has driven the barrier to the generation and sharing of information to nearly zero. Smartphones, laptops, tablets, Web cams, sensors, and other devices share text, imagery, video, and other data with a tap of a finger or through autonomous operation. Blogs, tweets, social media, and Web page updates, email and a host of other communication mechanisms course through the global Internet in torrents (no pun intended). Much, if not most, of the information found on the Internet seems to me to be beneficial; a harvest of human knowledge. But there are other consequences of the reduced threshold for access to the Internet.

The volume of information is mind-boggling. I recently read one estimate that 1.7 trillion images were taken (and many shared) in the past year. The Twittersphere is alive with vast numbers of brief tweets. The social media have captured audiences and contributors measured in the billions. Incentives to generate and share content

abound—some monetary, some for the sake of influence, some purely narcissistic, some to share beneficial knowledge, to list just a few. A serious problem is that the information comes in all qualities, from incalculably valuable to completely worthless and in some cases seriously damaging. Even setting aside malware, DDOS attacks, hacking and the like, we still have misinformation, disinformation, “fake news,” “post-truth alternate facts,” fraudulent propositions, and a raft of other execrable material often introduced cause deliberate harm to victims around the world. The vast choice of information available to readers and viewers leads to bubble/echo chamber effects that reinforce partisan views, prejudices, and other societal ills.

There are few international norms concerning content. Perhaps child pornography qualifies as one type of content widely agreed to be unacceptable and which should be filtered and removed from the Internet. There are national norms that vary from country to country regarding legitimate and illegitimate/illegal content. The result is a cacophony of fragmentation and misinformation that pollutes the vast majority of useful or at least innocuous content to be found on the Internet. The question before us is what to do about the bad stuff. It is irresponsible to

**The question
before us is
what to do about
the bad stuff.**

ignore it. It is impossible to filter in real time. YouTube alone gets 400 hours of video uploaded per minute (that is 16.7 days of a 24-hour television channel). The platforms that support content are challenged to cope with the scale of the problem. Unlike other media that have time and space limitations (page counts for newspapers and magazines; minutes for television and radio channels) making it more feasible to exercise editorial oversight, the Internet is limitless in time and space, for all practical purposes.

Moreover, automated algorithms are subject to error or can be misled by the action of botnets, for example, that pretend to be human users “voting” in favor of deliberate or accidental misinformation. Purely manual review of all the incoming content is infeasible. The consumers of this information might be able to use critical thinking to reject invalid content but that takes work and some people are often unwilling or unable to do that work. If we are to cope with this new environment, we are going to need new tools, better ways to validate sources of information and factual data, broader agreement on transnational norms all the while striving to preserve freedom of speech and freedom to hear, enshrined in the Universal Declaration of Human Rights.^a I hope our computer science community will find or invent ways to engage, using powerful computing, artificial intelligence, machine learning, and other tools to enable better quality assessment of the ocean of content contained in our growing online universe. □

^a <http://www.un.org/en/universal-declaration-human-rights/>

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

Copyright held by author.

Introducing ACM Transactions on Human-Robot Interaction

Now accepting submissions to ACM THRI

As of January 2018, the *Journal of Human-Robot Interaction* (JHRI) has become an ACM publication and has been 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 in the ACM Digital Library.

Co-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>



Association for
Computing Machinery



DOI:10.1145/3182625

Moshe Y. Vardi

A Declaration of the Dependence of Cyberspace

JOHN PERRY BARLOW, the famed founder of the Electronic Frontier Foundation, a digital-rights advocacy group, passed away on Feb. 6, 2018. In 1996, Barlow published “A Declaration of the Independence of Cyberspace.” It offered a rebuttal to Internet governance by national governments, opening with “Governments of the Industrial World, you weary giants of flesh and steel, I come from Cyberspace, the new home of Mind. On behalf of the future, I ask you of the past to leave us alone.”

It is hard to believe that such a naïve view of cyberspace was taken seriously just about 20 years ago, that people really believed simplistic statements such as “We believe that from ethics, enlightened self-interest, and the commonweal, our governance will emerge.” But we must remember that 20 years ago the Internet was indeed some kind of a New World, seemingly outside the shackling legacy of traditional governance. That was also before the Internet and the World Wide Web became dominated by giant corporations, and before Tim Berners-Lee, the recipient of the 2016 ACM Turing Award for inventing the World Wide Web, declared in 2017 that “The system is failing.”

What we have also learned in the past 20 years that while cyberspace may be indeed “the new home of Mind,” it is inextricably connected to the physical world. Indeed, the economic impact of the Web has been and continues to be profound. Newspapers are struggling to survive because advertising income, which has been their main source of revenue, has migrated to the Web, with Google and Facebook as the main beneficiaries. While e-commerce escalates, traditional retail outlets are shuttering

down daily, suffering from “retail apocalypse.”^a And while cyberattacks are now a daily occurrence, there are growing fears of a possible cyberattack that will knock out U.S. power grids.^b What happens in cyberspace does not stay in cyberspace!

To my mind, however, nothing epitomizes the hubris of the technoutopianists more than the idea of reinventing money. In October 2008, the mysterious Satoshi Nakamoto posted a white paper^c on the new domain of bitcoin.org, in which he asserted, “What is needed is an electronic payment system based on cryptographic proof instead of trust.” Bitcoin is based on a P2P network where transactions are cryptographically verified and recorded in a public distributed ledger based on blockchain, a distributed-consensus protocol. We now seem to be in the midst of a bitcoin mania, with the value of bitcoins gyrating wildly, making double-digit moves in a single week. There is also significant evidence^d that the bitcoin-exchange markets are being manipulated. But bitcoin has only been the first of dozens of cryptocurrencies, and initial coin offerings, which raise funds for issuing new cryptocurrencies, and are growing in popularity.

But even if bitcoin solved (quite imperfectly) the verifiability and distributed-trust issues, the idea of apolitical money is a dangerous fantasy. Verifiability and trust are only two requirements from a currency. Other requirements, which are intimately related, are value and supply. Central banks used to strive for a stable currency value. More

recently they have come to realize that a slightly depreciating currency value (about 2% per year) is better for economic growth. To achieve this, central bankers use a variety of sophisticated monetary tools to manage the money supply, taking into account a large number of economic indicators. This is an enormously complicated task challenging the best economic minds. In contrast, the supply of cryptocurrencies is a priori limited, and its gyrating value is determined by trading decisions made by “investors.” So the idea that apolitical cryptocurrencies will replace political money is a delusion.

Just like other speculative bubbles, the cryptocurrency bubble will also blow up at some point, though the timing is quite unpredictable. But the risk is not only to gullible speculators. As time goes on, cryptocurrencies get more enmeshed in our economic system and the risk of financial contagion grows. Financial contagion refers to the spread of market disturbances, typically on the downside, between different economic institutions and between different countries. The cryptocurrency bubble is, in my opinion, a growing systemic financial risk (and there are also the issues of susceptibility to cyberattacks and voracious energy appetite).

Just as you cannot separate the mind and the body, you cannot separate cyberspace and physical space. It is time to accept this dependence and act accordingly.

Follow me on Facebook, Google+, and Twitter. □

a <https://goo.gl/PEMP55>
b <https://goo.gl/esziCL>
c <https://goo.gl/55N41V>
d <https://goo.gl/dP1GTz>

Moshe Y. Vardi (vardi@cs.rice.edu) is the Karen Ostrum George Distinguished Service Professor in Computational Engineering and Director of the Ken Kennedy Institute for Information Technology at Rice University, Houston, TX, USA. He is the former Editor-in-Chief of *Communications*.

Copyright held by author.

Keep the ACM Code of Ethics As It Is

THE PROPOSED CHANGES to the ACM Code of Ethics and Professional Conduct, as discussed by Don Gotterbarn et al. in “ACM Code of Ethics: A Guide for Positive Action”¹ (Digital Edition, Jan. 2018), are generally misguided and should be rejected by the ACM membership. The changes attempt to, for example, create real obligations on members to enforce hiring quotas/priorities with debatable efficacy while ACM members are neither HR specialists nor psychologists; create “safe spaces for all people,” a counterproductive concept causing problems in a number of universities; counter harassment while not being lawyers or police officers; enforce privacy while not being lawyers; ensure “the public good” while not being elected leaders; encourage acceptance of “social responsibilities” while not defining them or being elected leaders or those charged with implementing government policy; and monitor computer systems integrated into society for “fair access” while not being lawyers or part of the C-suite.

ACM is a computing society, not a society of activists for social justice, community organizers, lawyers, police officers, or MBAs. The proposed changes add nothing related specifically to computing and far too much related to these other fields, and also fail to address, in any significant new way, probably the greatest ethical hole in computing today—security and hacking.

If the proposed revised Code is ever submitted to a vote by the membership, I will be voting against it and urge other members to do so as well.

Reference

1. <https://dl.acm.org/citation.cfm?id=3173016>

Alexander Simonelis, Montreal, Canada

Authors Respond:

ACM promotes ethical and social responsibility as key components of

professionalism. Computing professionals should engage thoughtfully and responsibly with the systems they create, maintain, and use. Lawyers, politicians, and other members of society do not always fully understand the complexity of modern sociotechnical systems; computing professionals can help this understanding. Humans understand such concepts as harm, dignity, safety, and well-being; computing professionals can apply them in their technical decisions. We invite Simonelis to read the Code and accompanying materials in more detail, as many of his claims, in our opinion, misread the Code. We also invite everyone else to read it, too; <https://ethics.acm.org/2018-code-draft-3/>

Catherine Flick, Leicester, U.K., and **Keith Miller**, St. Louis, MO, USA

‘Law-Governed Interaction’ for Decentralized Marketplaces

Given today’s sometimes gratuitous efforts toward centralized control over the Internet, I found it refreshing to read Hemang Subramanian’s article “Decentralized Blockchain-Based Electronic Marketplaces” (Jan. 2018) arguing that applications like electronic marketplaces and social networks would benefit from a decentralized implementation, describing a mechanism based on Bitcoin’s concept of blockchain imposing distributed protocols, or what is called “smart contracts” in this context.

Subramanian did not, however, mention the existence of a different, older, technique for implementing decentralized applications called “law-governed interaction,” or LGI, introduced in 1991 (under a different name) by Minsky.¹ It was implemented at Rutgers University some 10 years later and is still under development. LGI can be used to implement a range of decentralized applications, including decentralized marketplaces² and (in 2015) decentralized social networks, the very applications that attracted Subramanian’s interest.

It would have been instructive if Subramanian had, say, compared and contrasted LGI with blockchain-based mechanisms for enforcing distributed protocols, as they are two radically different mechanisms for achieving essentially the same objective.

References

1. Minsky, N.H. The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering* 17, 2 (1991), 183–195.
2. Serban, C., Chen, Y., Zhang, W., and Minsky, N. The concept of decentralized and secure electronic marketplace. *The Journal of Electronic Commerce Research* 8, 1–2 (June 2008), 79–101.

Naftaly Minsky, Edison, NJ, USA

Author Responds:

Comparing LGI and blockchain-based smart contracts would be a great idea, as Minsky says, as they are radically different approaches to decentralization. However, from an adoption standpoint what matters most is mass adoption at scale. For that to happen, the value created by decentralization would have to be shared among all users in some tangible way. Blockchain-based decentralization, in addition to ensuring secure low-cost distributed transactions, could make network effects fungible through the issuance of cryptocurrencies that can be exchanged for fiat currency; for example, Steem is a popular social network that issues virtual currency powered by the blockchain.

Hemang Subramanian, Miami, FL, USA

Scant Evidence for Spirits

Arthur Gardner’s letter to the editor “A Leap from Artificial to Intelligence” (Jan. 2018) on Carissa Schoenick et al.’s article “Moving Beyond the Turing Test with the Allen AI Scientific Challenge” (Sept. 2017) asked us to accept certain beliefs about artificial intelligence. Was he writing that all rational beings are necessarily spiritual? “That which actually knows, cares, and chooses is the spirit, something every human being has,” he said. And that all humans are rational? Why and how would someone (anyone) be convinced of such a hypothesis?

Not every human, to quote Gardner, “knows, cares, and chooses.” One might suspect that no human infant does, but may, in fact, learn and develop them over time.

What evidence for spirits? Would Gardner accept an argument that there are *no* spirits? If not, would this not be a rejection of the scientific method and evidence-based reasoning? Scientific hypotheses are based on experimental design. Valid experimental designs always allow for “falsifiability,” as argued by philosopher of science Karl Popper (1902–1994).

Falsifiability (sometimes called testability) is the capacity for some proposition, statement, theory, or hypothesis to be proven wrong. That capacity is an essential component of the scientific method and hypothesis testing. Through it, we say what we know because we test our beliefs using observation, not faith.

Humans are not rational by definition. They can think and behave rationally or not. Rational beings apply, explicitly or implicitly, the strategy of theoretical and practical rationality to the thoughts they accept and the actions they perform. A person who is not rational has beliefs that do not fully use the information he or she has.

“Man is a rational animal—so at least I have been told. Throughout a long life I have looked diligently for evidence in favour of this statement, but so far I have not had the good fortune to come across it,” said British philosopher Bertrand Russell (1872–1970), tongue firmly planted in cheek.

One might believe, without evidence, that “The leap from artificial to intelligence could indeed be infinite,” as Gardner claimed. However, every day newspapers in 22 countries are designed by my company’s AI-based software for classified pagination and display ad dummying. What was once done by rational, thinking human designers is now done by even more expert computer programs. And I started on this journey in 1973 by writing chess algorithms.

To replace humans, these programs have no need to know what a human is or to care.

Our “clever code” may just be our DNA that through long biological evo-

lution has developed into what we today call consciousness and rationality. Perhaps these are just emergent properties of a murmuration of neurons.

Richard J. Cichelli, Nazareth, PA, USA

Still Looking for Direction in Software Development

I have been in IT for 30 years, working on every kind of platform and thus feel qualified to address several points about systems development raised by Stephen J. Andriole in his Viewpoint “The Death of Big Software” (Dec. 2017). For example, I see in many current “agile” cloud-based projects a fundamental lack of direction. For projects that fail to perform as promised, the lack of a more in-depth requirements process can lead to missing critical integrations with other systems. I have personally seen at least a dozen projects spiral out of control and never reach a real live human user. For example, in 2016, I worked with a U.S. government agency on a very large project it had promised to deliver by 2020 but that failed a system test in the cloud because it could not meet its own integration and scalability goals. Even as the development team managed to occasionally pick off relatively minor user requests, it ignored the user-story requirements with deeper technical complexities, as in how to integrate with other systems. Lack of integration led to missed deadlines for delivering the key integrations by system test dates, as mandated by Article I, Section 2 of the United States Constitution.¹

As far as how an organization can get its data back if it moves from one cloud provider to another, the container “solution” might sound nice to users but can actually be worse than having table dumps from legacy systems. The lack of documentation around containers, both architecturally and with respect to how containers function within the workflow process and how the system will actually process data, makes designing for portability exceptionally difficult or impossible for IT managers to maintain during changes throughout the systems life cycle. Another challenge of working with containers involves

security analysts being able to perform proper system assessments. It is, in fact, some of the same micro services Andriole explored that can lead to security flaws that are then available for exploitation by aspiring hackers with a library of scripts that can be run against the containers and the host operating system.

Though I have great regard for cloud projects and the technology that allows faster and more-flexible solutions to address business needs, IT managers must make sure they do not lose the major benefits of enterprise resource planning products. I spent the 1990s moving from piece-meal systems to a system where a business user can track raw materials all the way to the end product and bought and sold with just a few clicks. I would hate to see IT managers lose that by going back to disparate processes lacking the transparent integration I know is possible.

Reference

1. Library of Congress. Article I—Legislative Department. U.S. Constitution; <https://www.congress.gov/content/conan/pdf/GPO-CONAN-2017-9-2.pdf>

Dan Lewis, Washington, D.C., USA

Author Responds:

The death of big software is attributable to failure, control, governance, cloud, and monolithic software, and I thank Lewis for addressing failure, cloud, and monolithic software. Cloud “containers” represent a first step toward hostage prevention. I agree that cloud security due diligence should always be aggressive. I also agree that integration is always important but that monolithic architectures do not guarantee integration (at the expense of flexibility) and that micro-services-based architectures can integrate and provide functional flexibility, with the right tools.

Stephen J. Andriole,
Villanova, PA, USA

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.



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:10.1145/3178118

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

The Costs and Pleasures of a Computer Science Teacher

Mark Guzdial considers the enormous opportunity costs of computer science teachers, while Bertrand Meyer ponders the pleasures of arguing with graduate students.



Mark Guzdial
The Real Costs of a Computer Science Teacher are Opportunity Costs, and Those Are Enormous

<http://bit.ly/2AvL2fz>

December 1, 2017

Imagine that you are an undergraduate who excels at science and mathematics. You could go to medical school and become a doctor, or you could become a teacher. Which would you choose?

If you are in the U.S., most students would not see these as comparable choices. The average salary for a general practitioner doctor in 2010 was \$161,000, and the average salary for a teacher was \$45,226. Why would you choose to make a third as much in salary? Even if you care deeply about education and contributing to society, the *opportunity cost* for yourself and your family is enormous. Meanwhile in Finland, the general practitioner makes \$68,000 and the teacher makes \$37,455. Teachers in Finland are not paid as much as doctors (<http://bit.ly/2m3ZnaK>), but Finnish teachers

make more than half of what doctors do. In Finland, the opportunity cost of becoming a teacher is not as great as in the U.S.

The real problem of getting enough computer science teachers is the opportunity cost. We are struggling with this cost at both the K-12 (primary and secondary school) level and in higher education.

I have been exchanging email recently with Michael Marder of UTeach at University of Texas at Austin (<http://bit.ly/2CKwScu>). UTeach (<https://uteach.utexas.edu/>) is an innovative and successful program that helps science, technology, engineering, and mathematics (STEM) undergraduates become teachers. They do not get a lot of computer science (CS) students who want to become CS teachers; CS is among the majors that provide the smallest number of future teachers. A 2011 U.K. report (<http://bit.ly/2CLviXF>) found that CS graduates are less likely to become teachers than other STEM graduates.

CS majors may be just as *interested* in becoming teachers. Why don't they? My guess is the perceived opportunity

cost. That may just be perception—the average starting salary for a certified teacher in Georgia is \$38,925 (<http://www.teachingdegree.org/georgia/salary/>), and the average starting salary for a new software developer in the U.S. (not comparing to exorbitant *possible* starting salaries) is \$55,000 (<http://bit.ly/2CFuQJL>). That's a big difference, but it's not the 3x differences of teachers vs. doctors.

We have a similar problem at the higher education level. The National Academies of Sciences, Engineering, and Medicine just released a report: *Assessing and Responding to the Growth of Computer Science Undergraduate Enrollments* (you can read it for free or buy a copy at <http://bit.ly/2CWttnt>), which describes the rapidly rising enrollments in CS (also described in the CRA *Generation CS* report, discussed in a previous blog at <http://bit.ly/2qiMahP>) and the efforts to manage them. The problem is basically too many students for too few teachers, and one reason for too few teachers is that computing Ph.D.'s are going into industry instead of academia.

Quoting from the report:

CS faculty hiring has become a significant challenge nationwide. The number of new CIS (computer and information science and support ervices) Ph.D.'s has increased by 21% from 2009 (1,567 Ph.D.'s) to 2015 (1,903 Ph.D.s), while CIS bachelor's degree production has increased by 74%. During that time, the percentage of new Ph.D.'s accepting jobs in industry has increased somewhat, from 45% to 57% according to the Taulbee survey. Today, academia does not necessarily look attractive to new Ph.D.'s: the funding situation is tight and uncertain; the funding expectation of a department may be perceived as unreasonably high; the class sizes are large and not every new hire is prepared to teach large classes and manage TAs effectively; and the balance between building a research program and meeting teaching obligations becomes more challenging. For the majority of new CS Ph.D.'s, the research environment in industry is currently more attractive.

The opportunity cost here influences the individual graduate's choice. The report describes new CS Ph.D. graduates looking at industry vs. academia, seeing the challenges of academia, and opting for industry. This has been described as the "eating the seed corn" problem (<http://bit.ly/2CtEo79>). (Eric Roberts has an origin story for the phrase at his website on the capacity crisis, at <http://stanford.io/2CXoQtx>.)

That is a huge problem, but a similar and less well-documented problem is when existing CS faculty take leaves to go to industry. I do not know of any measures of this, but it certainly happens a lot—existing CS faculty getting scooped up into industry. Perhaps the best-known example was when Uber "gutted" CMU's robotics lab (see the description at <http://bit.ly/2qw2wVh>). It happens far more often at the individual level. I know several robotics, AI, machine learning, and HCI researchers

who have been hired away on extended leaves into industry. Those are CS faculty not on hand to help carry the teaching load for "Generation CS."

Faculty do not have to leave campus to work with industry. Argo AI, for example, makes a point of funding university-based research, of keeping faculty on campus teaching the growing load of CS majors (<http://bit.ly/2CuHA2r>). Keeping the research on-campus also helps to fund graduate students (who may be future CS Ph.D.'s). There is likely an opportunity cost for Argo AI; by bringing the faculty off campus to Argo full-time, they would like get more research output. There is an associated opportunity cost for the faculty; going on leave and into industry would likely lead to greater pay.

On the other hand, industry that instead hires away the existing faculty pays a different opportunity cost. When the faculty goes on leave, universities have fewer faculty to prepare the next generation of software engineers. The biggest cost is on the non-CS major. Here at Georgia Tech and elsewhere, it is the non-CS majors who are losing the most access to CS classes because of too few teachers. We try hard to make sure that the CS majors get access to classes, but when the classes fill, it is the non-CS majors who lose out.

That is a real cost to industry. A recent report from Burning Glass (<http://bit.ly/2EdZvL5>) documents the large number of jobs that require CS skills, but not a CS major. When we have too few CS teachers, those non-CS majors suffer the most.

In the long run, which is more productive: Having CS faculty working full-time in industry today, or having a steady stream of well-prepared computer science graduates and non-CS majors with computer science skills for the future?

Comments

Great article. The first part of this highlights one of my major complaints about teacher's unions in the U.S. They push very hard to keep the pay for all teachers, regardless of subject taught, equal. In the case of CS, they are clearly hurting education by doing so.

I also have a comment on the opportunity cost analysis at the

beginning comparing doctors, teachers, and entry-level software developers. To do that comparison properly, you have to take into account the fact that doctors have to stay in school a long time. Doctors don't start making that kind of money until after four years of medical school and ~three years of residency. Both teaching and software development can get jobs right out of undergrad. So you have to factor in seven years of lost wages for the doctor. At that point, the salary for the teacher has risen a little, while that for the software developer has gone up quite a bit. So while I agree completely with the issue of opportunity cost, I think that this example needs more details to be complete.

—Mark Lewis

Thanks, Mark! Great point about the relationship between years of school and salary. I agree.

—Mark Guzdial



Bertrand Meyer

**Small and
Big Pleasures**

<http://bit.ly/2Cz77eQ>

December 19, 2017

One of the small pleasures of life is to win a technical argument with a graduate student. You feel good, as well you should. It is only human to want to be right. Besides, if you ended up being wrong all or most of the time, you should start questioning your sanity: Why are they the students and you the supervisor, rather than the other way around? (One of the most hypocritical lies in the world is the cliché "*I make sure to hire people who are smarter than I am.*" Sure. So obviously uttered—unless the person you are hiring is your successor—for the sole purpose of making you look whip-smart. If it were sincere, why then would you stay on?)

One of the big pleasures of life is to lose an argument with a graduate student. Then you have learned something.

Mark Guzdial is a professor in the College of Computing at the Georgia Institute of Technology in Atlanta, GA, USA. **Bertrand Meyer** is professor of Software Engineering at ETH Zurich, the Swiss Federal Institute of Technology; research professor at Innopolis University (Kazan, Russia), and chief architect of Eiffel Software (based in Goleta, CA, USA.).

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 _____

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

Science | DOI:10.1145/3178122

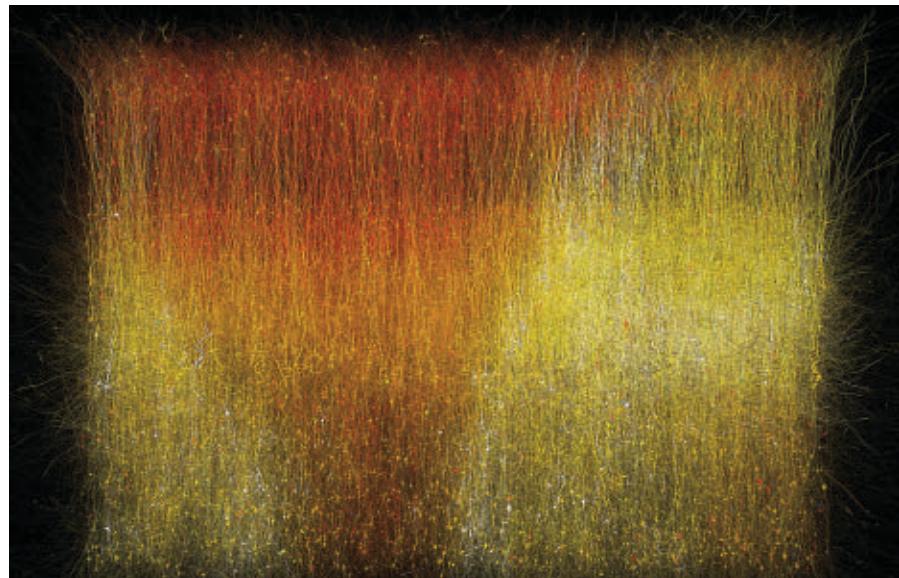
Samuel Greengard

In Pursuit of Virtual Life

Scientists are simulating biological organisms and replicating evolution in the lab. How far can they expand the boundaries of virtual life?

AT FIRST GLANCE, the creature known as *Caenorhabditis elegans*—commonly referred to as *C. elegans*, a type of roundworm—seems remarkably simple; it is comprised of only 959 cells and approximately 302 neurons. In contrast, the human body contains somewhere around 100 trillion cells and about 100 billion neurons in the brain. Yet decoding the genome for this worm and digitally reproducing it—something that could spur enormous advances in the understanding of life and how organisms work—is a challenge for the ages.

“The project will take years to complete. It involves enormous time and resources,” says Stephen Larson, project coordinator for the Open-Worm Foundation. Larson, a neuroscientist who is CEO of data software firm MetaCell, is not the only person focused on digitally reproducing life, or replicating evolution inside a computer. Researchers from a variety of fields are now attempting to decode worms, fly brains, and evolutionary processes in order to create virtual organisms and simulations of living creatures. It is safe to say that the field of executable biology—constructing computational models of biological systems—is coming to life.



Simulation of electrical activity in a “virtual brain slice” formed from seven unitary digital reconstructions of neocortical microcircuits.

Ultimately, this research could lead to a far greater understanding of how neurons fire and brains and entire organisms function. This knowledge would likely lead to new therapies and drugs for treating sickness and disease, but it could also produce new biofuels, as well as entirely new computing frameworks. It also raises questions about what constitutes life and whether living things can be engineered inside a computer.

Says Larson: “Understanding how organisms function would unlock many of the secrets of nature and change the way we view and interact with the world.”

Beyond Biology

The idea of developing virtual organisms and simulating physical systems through computing is nothing new. In the late 1940s, John von Neumann began exploring the concept of creating a computer virus modeled after a biologi-

cal virus; he eventually developed the first self-replicating program. In 1970, mathematician John Horton Conway introduced a cellular automation system called *Conway's Game of Life*, in which a person configures a set of circles, then the computer embarks on a rudimentary evolutionary process.

By the 1990s, a number of researchers had begun to explore the idea of producing digital representations of biological creatures.

For scientists, the idea of creating virtual life and artificial worlds inside a computer is rooted in practicality: it makes possible the study of the genetic information of an organism, or the creation of a virtual space to study how evolution and adaption take place. "Researchers can run thousands and thousands of replicates simultaneously. Every computer is essentially a Petri dish," explains Christoph Adami, professor of microbiology and molecular genetics at Michigan State University. This approach also allows researchers to isolate specific components, including genetic coding, and "very carefully tease apart the different elements that go into the evolutionary process," he says.

OpenWorm is an example of how this new frontier of biology and computing is unfolding. So far, "several hundred people" have contributed to the project in some way. This includes computer scientists, mathematicians, biologists, and experts in neuroscience. Among the core participants are more than a dozen academic and research luminaries, including *C. elegans* biologists Sreekanth Chalasani at the Salk Institute, Michael Francis at the University of Massachusetts Medical School, William Schafer at University of Cambridge, and Andrew Leifer at Princeton University. In addition, the organization has received computing input from the likes of Neta Cohen at Leeds University and Christian Grove at the California Institute of Technology (CalTech).

The OpenWorm project is now nearly seven years old. Larson estimates the project is 80% of the way toward achieving its first goal: assembling a digital model of the worm that allows researchers to simulate movement through simulated viscous fluids. The team hopes to achieve this milestone by late this year. This has involved mapping cells and

Mountains of data produce incremental gains, and coordinating all the research groups and silos is a complex endeavor.

functions in the worm's body, developing software to run simulations, building a digital model of *C. elegans*, and constructing an algorithm that simulates the worm's muscle movements—including how electrical signals travel through its brain and nervous system.

So far, researchers at Caltech have developed the OpenWorm Browser, which relies on a Web or iOS interface to display a three-dimensional anatomical model and actions for *C. elegans*. The browser displays different layers, including the skin, alimentary system, nervous system, reproductive system, and body wall muscles. In addition, a program called *Sibernet* uses a C++ algorithm to model and simulate contractile matter and membranes within the muscle tissue of the *C. elegans*. Another platform, *Gepetto*, provides an open source Web-based neuroscience simulation and visualization environment that simulates complex biological systems and their surrounding environment using multiple algorithms.

Not surprisingly, the data processing challenges related to OpenWorm and developing a life-like digital model are enormous; the overall task of understanding things like synthesis, reproduction, and digestion will likely take several more years. For now, researchers rely on a combination of classical mathematical and analytics tools along with machine learning to decode functions at the level of ion channels and cells. "Cells have a lot of extra machinery in them that is difficult to detect, and many of these activities and processes are completely ignored by artificial neural nets," Larson says. Simply put: mountains of data produce incremental gains, and coordinating all the research groups

and silos is a complex endeavor. Ultimately, "We may need to get to a new type of computing process to understand the exotic dynamics of natural neural systems," he says.

Cracking the Code

The OpenWorm project is one of several current attempts to unravel the mysteries of living things.

For example, Virtual Fly Brain—a joint effort involving the University of Edinburgh, University of Cambridge, MRC Laboratory of Molecular Biology, Cambridge, and the European Bioinformatics Institute—is mapping the physiology of the household fly.

In 2012, Jonathan Karr at the Institute for Genomics & Multiscale Biology Institute at the Mt. Sinai School of Medicine in New York City assembled the first whole-cell model of *Mycoplasma genitalium*, a pathogenic bacterium that resides in humans. The model succeeded in predicting the viability of cells after genetic mutations.

In 2016, Stanford University bioengineering professor Markus W. Covert and a team of researchers developed a whole-cell computational model; they used detailed information from more than 900 scientific journals to gain insights into previously unobserved cellular behaviors.

There's also the work of Henry Markram, a professor of neuroscience at the École Polytechnique Fédérale de Lausanne in Switzerland, director of that institution's Laboratory of Neural Microcircuitry, and founder and director of the Swiss Blue Brain Project national brain initiative. His research has focused on synaptic plasticity and the microcircuitry of the neocortex. In 2005, he launched the initiative in order to reconstruct and simulate the mammalian brain, starting with the rodent neocortical column. Markram and fellow researchers are now attempting to reverse-engineer the circuitry of the brain—something that could radically redefine health and medicine.

Make no mistake, these projects extend far beyond a basic understanding of physiological mechanisms. An organism's behavior is affected by numerous factors, ranging from its environment to its genetics. This means that even when scientists decode the genome of a creature such as *C. elegans*, it remains

incredibly challenging to understand how cells function alone and together, and how they interact with the environment to adapt, adjust, and evolve. “Achieving a complete understanding of a worm requires incredible resources. Understanding the mechanisms in more advanced lifeforms is still very far off into the future,” says Herbert Sauro, associate professor of bioengineering at the University of Washington.

“It’s painstaking and arduous work to put all the pieces together,” says Alexander Hoffman, professor of immunology and microbiology at the University of California, Los Angeles. “It’s necessary to pull together research from a very large pool of existing literature, code all the information in a set of equations and parameters, and then work with computing software to relate model simulations to all the data. The problem for now is that there’s often not enough existing knowledge to deliver an accurate simulation of a phenotype—and so you wind up with gaps in knowledge that require further experimentation.”

Mind Games

A primary goal of these projects, and executable biology in general, is to produce reliable computer models that ultimately can be used to understand the behavior of cancer cells and address other debilitating or life-threatening diseases, ranging from multiple sclerosis and amyotrophic lateral sclerosis to heart disease and arthritis, says Sauro. “Understanding the internal machinery of cells and how they successfully orchestrate cellular remodeling in a way that doesn’t harm them could lead to faster and better ways to develop therapies and drugs.”

A deeper understanding of cellular activity also could help researchers engineer and reengineer organisms to produce biofuels and other chemical substances, or to produce entirely new categories and types of antibiotics and other medicines.

This field of research may also have enormous implications for computing, Larson says. Today, computational neuroscience attempts to faithfully reproduce the activity of neurons. However, neural nets do not exactly replicate the actions and behaviors of biological cells and neurons. “It’s not obvious what information processing neurons

are doing when you consider them as biological cells. There are more exotic dynamics taking place, but we cannot see them,” he says. However, “It may be possible to develop more advanced types of computing systems based on biology. It appears that there is more we can do with neural nets than we have been doing with deep learning.”

Of course, the ultimate questions for researchers and society are where will all of this lead, and how exactly do we define life? At some point, digital code could replicate biological code for entire creatures, and researchers in synthetic biology might compile code to engineer new types of organisms—or autonomous devices, such as robots, that use biological models to think. This may bring the world to the highly discussed state of “singularity,” where intelligence becomes increasingly non-biological and humans transcend their biological origins.

Concludes Sauro: “In the future, we could see very different definitions of life. Once you start creating and evolving life-like behaviors in computer code or in synthetic biological systems and then applying them to the physical world, it’s possible to produce a very different reality.” □

Further Reading

Adami, C., Hintze, A., Edlund, J.A., Olson, R.S., Knoester, D.B., Schosau, J., Albantakis, L., Tehrani-Saleh, A., Kvam, P., Sheneman, L., Goldsby, H., and Bohm, C.

Markov Brains: A Technical Introduction. Sept. 17, 2017. <https://arxiv.org/pdf/1709.05601.pdf>.

Basak, S., Behar, M., and Hoffmann, A. Lessons from Mathematically Modeling the NF_KB Pathway. *Immunological Reviews*, 2012. 246, pp.221-38. PMID: 22435558 PMCID: PMC3343698.

Palyanov, A., Khayrulin, S., and Larson, S.D. Application of smoothed particle hydrodynamics to modeling mechanisms of biological tissue. *Advances in Engineering Software*. 2016. 98, 1-11. doi: 10.1016/j.advengsoft.2016.03.002.

Misirli, G., Hallinan, J., Pocock, M., Lord, P., McLaughlin, J.A., Sauro, H., and Wipat, A. Data Integration and Mining for Synthetic Biology Design. *ACS Synthetic Biology*, April 25, 2016. Vol. 5, Issue 10, pp. 1086-1097. <http://eprints.keele.ac.uk/3637/>.

Samuel Greengard is an author and journalist based in West Linn, OR, USA.

© 2018 ACM 0001-0782/18/3 \$15.00

ACM Member News

UNDERSTANDING THE POWER COMPUTING CAN PROVIDE



Allison Druin is the inaugural Associate Provost for Research & Strategic Partnerships at the Pratt Institute in Brooklyn, NY. Her early education as a graphic designer, combined with her technology background, provides her with a skill set well suited for this new role.

Druin earned her undergraduate degree in Graphic Design at the Rhode Island School of Design, her Master's degree in Media Arts & Sciences from the Massachusetts Institute of Technology (MIT) Media Lab, and her Ph.D. from the University of New Mexico's College of Education. During her time at MIT, developing new technologies for children became Druin's personal research focus. “When you want to make new technology for children, there is no straight line from here to there,” Druin explains.

During two decades as a professor at the University of Maryland (UM), Druin served in numerous roles, including lab director, associate dean, and chief futurist. She describes her experience as giving voice to users in the innovation process, and in developing an understanding of the impact of new innovations.

In 2015, Druin took leave from UM to work as Special Advisor for National Digital Strategy for the National Park Service, where she led strategic planning to bring digital experiences to park visitors, and to also enhance digital preservation.

In her new role, Druin will lead the initiative to expand research within Pratt, as well as with the Institute's external partners. “You don't have to be a computer scientist to understand the power that computing can bring to the world,” she says.

—John Delaney

The Construction Industry in the 21st Century

Three-dimensional printing and other new technologies are revitalizing the business of building buildings.

THE CONSTRUCTION OF New York's Empire State Building is often seen as the figurative and literal pinnacle of construction efficiency, rising 1,250 feet and 102 stories from the ground to its rooftop spire in just over 13 months' time, at a human cost of just five lives. Indeed, most of today's construction projects would be lucky to come close to that level of speed, regardless of the building's size. While the construction industry traditionally has been slow to change the way it operates, several new technologies are poised to usher in a new era of faster and more automated construction practices.

Three-dimensional (3D) printing is among the key technologies that are expected to change the way structures are built in the future, as construction engineers and contractors seek methods for completing buildings more quickly, more efficiently, and, in many cases, with a greater attention paid to sustainability. Large printers that can print construction materials such as foam or concrete into specific shapes can drastically speed up the creation of walls, decorative or ornamental pieces, and even certain structural elements. Furthermore, in some scenarios, custom-built or unique items can be created onsite or in a factory, at a much lower cost than by using traditional, one-off casting techniques.

"If you can focus on [printing] the more labor-intensive components of the structure, then the productivity will increase," says Pelin Gultekin-Bicer, Building Information Modeling (BIM)/Virtual Design and Construction (VDC) project manager at VIATEchnik LLC, a construction and engineering services firm based in Chicago. In addition, she says, "if you can integrate the 3D printers onsite, then you can improve the



A concrete villa in Binzhou, China, that was produced by a 150-meter-tall 3D printer.

productivity, as [the construction schedule] will be more predictable."

The most common commercial use of 3D printing today is to create the molds that are used to cast concrete panels for use in a building or tunnel, which is how the technology is being used by Australia's Laing O'Rourke, a multinational construction company. Laing O'Rourke's FreeFAB technology employs a 100x25x15-foot robotic 3D printer that prints molds measuring 6x4.5 feet from a specially designed wax. These molds are used to cast large concrete panels at an offsite factory that are unique, in terms of size or shape, and likely would require a large traditional wood or polystyrene mold to be created for each panel.

According to the company, the FreeFAB technology is more eco-friendly and efficient than creating conventional wood or polystyrene molds, be-

cause it can print to an exact shape with micron-level accuracy, and the wax molds can be melted down for reuse again and again.

The completed panels are shipped and installed into the passenger tunnels of London's Crossrail railway construction project. Crossrail will be a high-frequency, high-capacity railway serving London and the Southeast U.K. According to James Gardiner, CTO and cofounder of FreeFAB, a spinout of Laing O'Rourke, utilizing 3D printing to create the molds, rather than the panels themselves, incorporates all of the benefits (speed and quick customization) of 3D printing, while minimizing the technology's weaknesses.

Gardiner notes that 3D concrete printers require very specific temperature and humidity levels in order for concrete to set properly. In addition, if a 3D printer pauses or stops, the homo-

geneity of each layer of concrete, and the bonds between each layer, will be compromised. This introduces questions about whether the material and building process will be able to be certified by local building authorities, given that any material used in construction must meet load-handling, weatherproofing, and other specific building code issues.

"[Research universities] are starting to focus on the characterization of 3D materials, trying to develop specific materials that are particularly well-suited for 3D printing and understanding those already used," Gardiner says. "But I'd say that it's still got a little way to go."

Other organizations, however, have taken to using 3D printing to construct entire buildings. Chinese company WinSun claims that in 2008, it printed an entire house using 3D printing technology in just two days. The company has since 3D-printed larger structures, such as a five-story section of a city block, as well as one of its own manufacturing plants in Suzhou Industrial Park. Each of these projects took about a month, far less time than would be required with traditional construction techniques.

According to industry professionals, however, WinSun's 3D-printed building technology still requires that the individual walls, floors, and roof be fastened together using traditional methods, including bolting walls and roofs together, and much of the internal elements of a house (ductwork, electrical conduit, plumbing, and other finishes) must still be installed, rather than printed.

"They're not printing an entire six-story apartment building in one go," explains Casey Mahon, digital practice manager at Carrier Johnson, an architectural, interior design, and branding practice based in San Diego, CA. "They're printing it in pieces. Those pieces are coming out to the job site, and they're getting assembled. Then they're getting clad with some tile or stone on the exterior. It's rare that you ever see an interior photograph of one of those projects. When you do, you can see that, it's clear, all the conduit is surface-run, all the fixtures are surface-mounted fixtures."

Meanwhile, researchers in France from the University of Nantes,

Nantes Métropole, Nantes Métropole Habitat (NMH), and Ouest Valorisation, with help from teams at the Nantes Digital Science Laboratory and the Institute of Research in Civil and Mechanical Engineering, are working to create an industrial 3D printer that will be able to build a demonstration house in only a few days. Called BatiPrint3D, the printer was completed in the fall of 2017. The device prints the home in three layers, including two foam layers for insulation, and a third concrete layer. Of course, the technology is simply being demonstrated, and is not yet ready for mass commercialization.

Similar to the WinSun house, a real functioning house would need to have its infrastructure and finishes added after printing, thereby putting real construction times in line with those of traditionally built houses.

That is why it is most likely that 3D printing will be primarily used either offsite, to facilitate the faster creation of molds to create one-off castings of

unique panels or parts, or will be deployed onsite to print unique or bespoke architectural or decorative pieces of a structure, rather than entire houses, in the near future. By addressing the labor-intensive pieces via onsite printing, 3D printing likely will improve the predictability of scheduling of material delivery and improve worker productivity, which together are responsible for much of the costs related to construction.

"The biggest problem of the construction industry is the variance in schedule and cost estimation," says Gultekin-Bicer, explaining that these include material costs, shipping costs, and labor costs. "Cost estimations are derived from the schedule, so if you extend the schedule of the building, you are also extending the cost," she says, noting that the variability in the schedule is largely due to the human labor factor. "If you can integrate the 3D printers onsite, then we can improve the productivity, as [the schedule] will be more predictable."

Autonomous Vehicles Help Build the Future

It is likely autonomous vehicles that can be operated within an enclosed setting, such as vehicles used on construction sites, will gain traction before self-driving cars do.

San Francisco-based Built Robotics is equipping construction vehicles with technology that allows them to do work that is dangerous, repetitive, or both, with little human intervention. Late last year, the company launched an autonomous track loader (ATL) that uses a combination of LIDAR sensors, inertial measurement units, and global positioning system (GPS) technology to handle basic construction site tasks, such as digging foundation holes.

Caterpillar, one of the world's largest manufacturers of heavy equipment, has deployed more than 100 autonomous haul trucks to mines throughout the world. According to the company, Caterpillar worked with Blacksburg, VA-based Torc Robotics for a decade to develop its RemoteTask skid steer remote control system. Caterpillar also partnered with Torc to develop a system (due for release early next year) in which a Komatsu 930E haul truck can be operated by the Caterpillar autonomous haul truck system.

"I think the biggest challenge to autonomy is going to be on the security side," says Bob Schena, chairman, CEO, and co-founder of Rajant Corp., a wireless mesh networking company that works with construction equipment manufacturers. "If governments and regulators believe that equipment could be hacked, taken over, and controlled by bad actors, they're not going to allow autonomy. So, this is an issue that will have to be addressed, and we're very involved in that."

"Right now, we are working with construction companies that are going to use our solutions in order to securely connect their equipment," says Moshe Shlisel, CEO of Ramle, Israel-based GuardKnox. Shlisel says many construction companies are using relatively weak connectivity software to actively monitor and control heavy equipment, such as cranes 200 feet tall.

"The manufacturer of the construction equipment would like to be able to monitor and to predict some malfunctions [by keeping an active and constant wireless connection to the crane], Shlisel says. "But you have to do that securely, otherwise you're having a connected motor or a connected crane that is exposed to every hacker that, just for fun, says to himself, 'well, let's see if I can control this crane remotely.'" —K.K.



Advertise with ACM!

Reach the innovators and thought leaders working at the cutting edge of computing and information technology through ACM's magazines, websites and newsletters.



Request a media kit with specifications and pricing:

Ilia Rodriguez

+1 212-626-0686

acmmEDIASales@acm.org

acm

media

A bricklaying robot called SAM100 can lay 3,000 bricks a day, six times as many as a typical human bricklayer.

In the future, Gardiner notes, 3D printing will allow for a more streamlined use of materials, thanks to the ability to print both structural and decorative pieces of a building that place material only where needed for maximum strength, energy efficiency, or form, or to help them fit into unusual or tight site constraints.

"At the moment, if you build a concrete wall or a brick wall, you're putting the same bulk material uniformly, regardless of where the stresses are on that particular element," Gardiner says, observing that 3D printing will allow architects to design building components that put material only where they are absolutely needed, rather than simply adhering to a mass-produced shape.

It is not just 3D printing that is poised to disrupt the construction industry. Created by New York-based company Construction Robotics, a bricklaying robot called SAM100 can lay 3,000 bricks per day, effectively multiplying a typical human bricklayer's productivity of about 500 bricks per day by six. The SAM100 (whose name stands for "semi-automated mason") uses a conveyor belt, robotic arm, and concrete pump to lay bricks. The robot's software ensures the robot can quickly choose between types of bricks, quickly lay bricks in complicated patterns, and strictly adhere to the building plan. However, the technology still requires a human operator to smooth the concrete before placing additional layers of bricks.

"Automation is popular in a controlled environment, but construction sites are not very controlled environments," says Zak Podkaminer, a marketing executive with Victor, NY-based Construction Robotics. He cites weather (such as rain, snow, and humidity),

space constraints, and the changing layout of the site as construction progresses as factors that can make a construction site less than ideal for automation. "There's a shortage of skilled workers that are going into the construction trades, so what the SAM does is allow each worker to be more productive."

Podkaminer notes that while the robot is expensive, there has been a lot of interest from contractors and masons in renting the robot, which allows it to be used by contractors that normally wouldn't have enough work to amortize its approximately \$500,000 purchase price over a longer time horizon.

Still, despite the increasing use of automation and 3D printing on the job site, the construction industry is notoriously slow when it comes to adopting new technology. This is largely due to the highly regulated nature of construction, as well as the high cost of adopting new technology, as Gardiner notes that the 3D printer used to create the massive molds can cost \$1 million.

"There are inexpensive 3D construction [concrete] printers around," Gardiner says, "but these machines are generally trading off accuracy and reliability to achieve their low cost. So, one of the things that I see is that a good concrete printer, a good construction 3D printer, will need to be highly reliable, fast, and accurate. And the problem with that is that a machine that has those things is generally more expensive." C

Further Reading

Gardiner, J.

2011, Exploring the Emerging Design Territory of Construction 3D printing – Project Led Architectural Research, Architecture & Design, RMIT University. <https://researchbank.rmit.edu.au/view/rmit:160277/Gardiner.pdf>

Building Codes, Underwriters Laboratories <https://www.ul.com/code-authorities/building-code/>

Laing O'Rourke's FreeFAB Technology https://www.youtube.com/watch?v=alAHME_A8VY

SAM100 OS 2.0

<https://www.youtube.com/watch?v=tK9oBQyR9KY>

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

The State of Fakery

How digital media could be authenticated, from computational, legal, and ethical points of view.

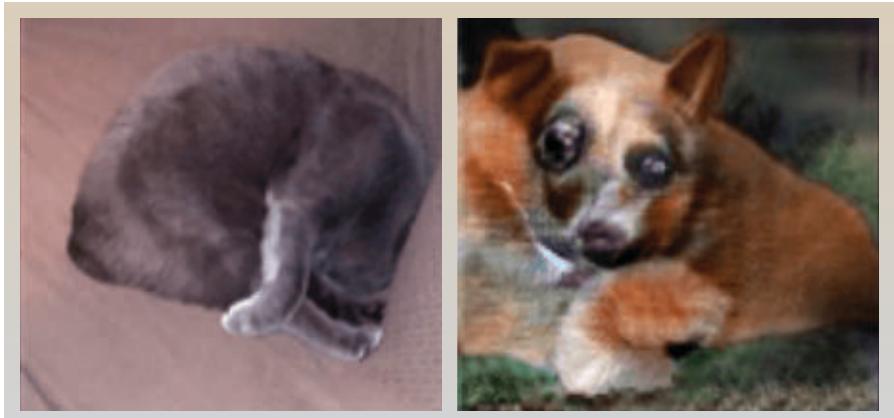
BACK IN 1999, Hany Farid was finishing his postdoctoral work at the Massachusetts Institute of Technology (MIT) and was in a library when he stumbled on a book called *The Federal Rules of Evidence*. The book caught his eye, and Farid opened to a random page, on which was a section entitled “Introducing Photos into a Court of Law as Evidence.” Since he was interested in photography, Farid wondered what those rules were.

While Farid was not surprised to learn that a 35mm negative is considered admissible as evidence, he was surprised when he read that then-new digital media would be treated the same way. “To put a 35mm file and a digital file on equal footing with respect to reliability of evidence seemed problematic to me,” says Farid, now a computer science professor at Dartmouth College and a leading expert on digital forensics. “Anyone could see where the trends were going.”

That led Farid on a two-decades-long journey to consider about how digital media could be authenticated from the computational, legal, and ethical points of view. He and others have their work cut out for them; there are dozens of ways a digital image can be manipulated, from doing something as simple as cropping or lightening it, to something more nefarious.

As fake news dominates headlines and the use of artificial intelligence (AI) to alter images, video, or photographs is rampant, media outlets, political campaigns, ecommerce sites, and even legal proceedings are being called into question for the work they generate. This has led to various efforts in government, academia, and technological realms to help identify such fakery.

“More and more, we’re living in a digital world where that underlying digital media can be manipulated and altered, and the ability to authenticate is incredibly important,” says



A real dog (left), and an image of a dog created by a deep convolutional generative adversarial network (GAN) algorithm.

Farid. Videos can go viral in a matter of minutes; coupled with the pace of technological advance and the ability to easily deceive someone online, how is it possible to trust what we’re seeing coming out in the world?

It is a troubling question, and no one, it seems, is immune from being targeted. Virgin Group founder Sir Richard Branson revealed he has been the target of scams using his image to impersonate him. In a blog post in January 2017, Branson noted that “the platforms where the fake stories are spreading need to take re-

sponsibility... and do more to prevent this dangerous practice.”

People need to be skeptical of what they see and hear, says David Schubmehl, research director, Cognitive/AI Systems and Content Analytics at market research firm IDC.

“People have to get used to the idea that images, voice recordings, video, and any other forms of media that can be represented as digital data can be manipulated and changed in one or more ways through the use of software,” Schubmehl says.

Because machine learning is becoming so prevalent, experts say we need ways to improve its ability to spot deception. For example, Schubmehl says, “Today, researchers are experimenting with generative adversarial networks (GANs) that can be used to combine two different types of images or video together to create a merged third type of video.”

The idea behind GANs is to have two neural networks, one that acts as a “discriminator” and the other a “generator,” which compete against each other to build the best algorithm for solving a problem. The generator network uses feedback it receives from the discriminator to learn to produce convincing images that can’t be distinguished from real images; this helps it get better at detecting something false.

Images, voice recordings, video, and other forms of media can be changed through the use of software, so we need better ways to spot deception.

Of course, image altering has its (legal) benefits. It has allowed the movie industry to produce spectacular action movies, since the vast majority of that action is computer-generated content, Farid points out. On the consumer side, image altering lets people create aesthetically pleasing photos, so everyone looks good in the same photo.

Schubmehl agrees. "Hollywood has been creating fake worlds for people for decades and now regular people can do it as well," he says. "There's a tremendous use for tools like Photoshop to create exactly the right type of image that someone wants for whatever purposes."

Whether manipulated images should be identified as such is the subject of much debate. While it would seem to be a no-brainer when it comes to their use in legal cases and photo-journalism, there are mixed opinions about the use of manipulated images in industries such as fashion, entertainment, and advertising. France recently passed a law stipulating that any image showing a model whose appearance has been altered must feature a clear and prominent disclaimer label to indicate this is the case, notes Sophie Nightingale, a postdoctoral teaching associate at Royal Holloway University of London. Those who do not comply with the law are subject to a fine of 30% of the advertising cost.

France enacted legislation requiring images showing models whose appearances have been altered to be clearly, prominently labeled to that effect.

"Perhaps not too surprisingly, advertisers and publishers continue to resist such legislation and criticize the limitations it places on free expression and artistic freedom," says Nightingale, who completed image manipulation studies as part of her Ph.D. at the University of Warwick in Coventry, U.K. She adds that "many photographers believe that the use of image manipulation techniques is a positive thing that allows creative freedom; in fact, some suggest the ability to manipulate images makes a photographer more akin to a painter who takes something that is real and puts their own artistic spin on it."

One tricky thing is that "manipulated" is not an easy word to define,

observes Farid, and there are a lot of gray areas. He advocates that publishing and media outlets, as well as courts of law and scientific journals, should adhere to a policy of "you don't have to tag the images, you simply have to show me the original." This, he says, "keeps people honest." That approach "allows we, the consumers, to make the determination, and bypasses the complexity of defining what's appropriate and what isn't," Farid says.

Right now, we are unable to know for sure when a photo has been altered when sophisticated manipulations are being used, says Nightingale. She adds, however, that there are signs of image manipulation that can be identified.

"Computer scientists working in digital forensics and image analysis have developed a suite of programs that detect inconsistencies in the image, perhaps in the lighting," she says. Her work includes conducting research to see whether people can make use of these types of inconsistencies to help identify forgeries.

Other more general tips Nightingale suggests for spotting fake photos include using reverse image searches to find the image source, looking for repeating patterns in the image (since repetition might be a sign that something has been cloned) and checking the metadata, which provides details such as the date and time the photo

Milestones

BBVA Recognizes Turing Laureates

The BBVA Foundation, a Spanish organization that promotes research, advanced training, and the transmission of knowledge to society, has given its 10th Frontiers of Knowledge Award in the Information and Communication Technologies category to ACM A.M. Turing Award recipients Shafi Goldwasser, Silvio Micali, Ronald Rivest, and Adi Shamir for their "fundamental contributions to modern cryptology, an area of a tremendous impact on our everyday life," in the words of the jury's citation.

"Their advanced crypto-protocols enable the safe

and secure transmission of electronic data, ranging from email to financial transactions. In addition, their work provides the underpinning for digital signatures, blockchains, and cryptocurrencies."

The work of Goldwasser, Micali, Rivest, and Shamir, the citation adds, "is crucial to the fabric of our connected digital society. Every time we log in to social media, purchase goods online, or vote or sign electronically, we leverage the technology developed by their research."

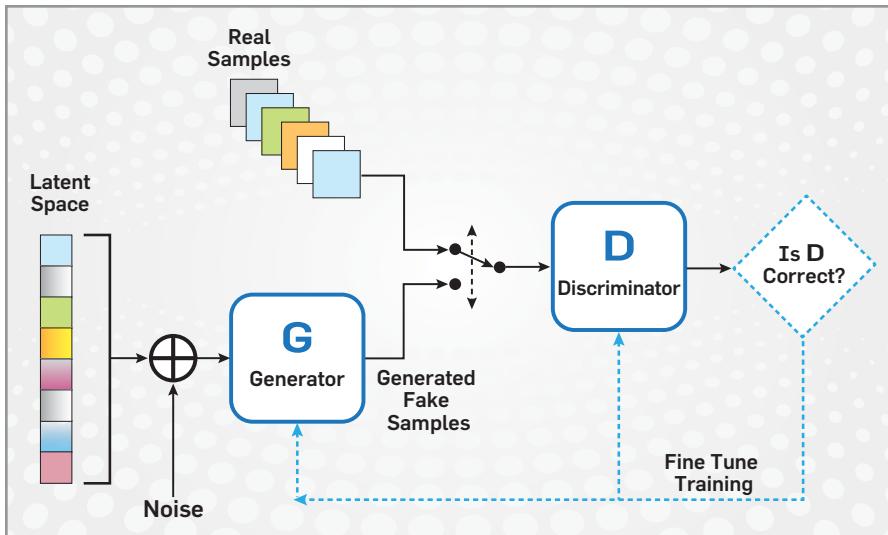
In 1978, Shamir and Rivest, together with Leonard Adleman, created the RSA algorithm, the "first of

the secure protocols that defined the face of modern cryptography," as the jury terms it. RSA is a "public-key" encryption system because each user has two keys: a public key, used to encrypt the message; and another known only to the receiver. The encryption process is based on a mathematical problem intractable for today's computers without the aid of the other, private key. RSA is still a widely used protocol, particularly in combination with other techniques.

Goldwasser and Micali in 1982 were working on a doctorate course at the University of California,

Berkeley, when they embarked on a collaboration whose first big result would lay the theoretical foundations of the field—the mathematical demonstration of when an encryption method is genuinely unbreakable.

The BBVA jury said Goldwasser and Micali, together and separately, "have expanded the scope of cryptography beyond its traditional goal of secure communication," with developments that have helped build today's flourishing digital society by allowing users to collaborate, share information, and shop online without sacrificing security.



Architecture of a generative adversarial network.

was taken, camera settings used, and location coordinates.

While digital forensic techniques are a promising way to check the authenticity of photos, for now “using these techniques requires an expert and can be time-consuming,” Nightingale says. “What’s more, they don’t 100% guarantee that a photo is real or fake. That said, digital forensic techniques and our work, which is trying to improve people’s ability to spot fake images, does at least make it more difficult for forgers to fool people.”

Farid has developed several techniques for determining whether an image has been manipulated. One method looks at whether a JPEG image has been compressed more than once. Another technique detects image cloning, which is done when trying to remove something from an image, he says. In addition, Schubmehl cites the development of machine learning algorithms by researchers at New York University to spot counterfeit items.

The mission of a five-year U.S. Defense Advanced Research Projects Agency (DARPA) program called MediFor (media forensics) is to use digital forensics techniques to build an automated system that can accurately analyze hundreds of thousands of images a day, says Farid, who is participating in the program. “We’re now in the early days of figuring out how to scale [the system] so we can do things quickly and accurately to stop the spread of viral content that is fake

or has been manipulated,” he says. “The stakes can be very, very high, and that’s something we have to worry a great deal about.”

That is because a growing number of AI tools are increasing the ability for forgery to flourish, regardless of how they are being used. In 2016, Adobe announced VoCo (voice conversion), essentially a “Photoshop of speech” tool that lets a user edit recorded speech to replicate and alter voices.

Face2Face is an AI-powered tool that can do real-time video reenactment. The technology lets a user “animate the facial expressions of the target video by a source actor and re-render the manipulated output video in a photorealistic fashion,” according to its creators at the University of Erlangen-Nuremberg, the Max Planck Institute for Informatics, and Stanford University. When someone moves their mouth and makes facial expressions, those movements and expressions will be tracked and then translated onto someone else’s face, making it appear that the target person is making those exact movements.

On the flip side is software that helps users take preventative measures against being duped. One is an AI tool called Scarlett that was recently introduced by adult dating site SaucyDates, with the goal of reducing fraud and scams in the dating industry. Scarlett acts as a virtual assistant and as people are having live conversations, it scores users; when the score reaches a threshold, it is flagged and

read by a moderator. To protect the privacy of the conversation, the moderator can only read the suspected fraudster’s messages, explains David Minns, founder and CEO of software developer DM Cubed, which developed the SaucyDates tool. He adds the AI tool also warns the potential victim of fraudulent content.

Farid says we should absolutely be alarmed by the growth of software that enables digital media to be manipulated into fakes for nefarious purposes.

“There’s no question that from the field of computer vision to computational photography to computer graphics to software that is commercially available, we can continue to be able to manipulate digital content in ways that were unimaginable a few years ago,” he says. “And that trend is not going away.” □

Further Reading

Physical Unclonable Functions and Applications: A Tutorial. *Proceedings of the IEEE*, Volume: 102, Issue: 8, Aug. 2014

Pappu, S.R.

Physical One-Way Functions. Pappu, Ravikanth. 2001. Massachusetts Institute of Technology, <http://cba.mit.edu/docs/theses/01.03.pappuphd.powf.pdf>

Thies, J., Zollhofer M., Samminger, M., Theobalt, C., and Niessner, M.

Face2Face: Real-time Face Capture and Reenactment of RGB Videos, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) http://www.graphics.stanford.edu/~niessner/papers/2016/1facetoface_thies2016face.pdf

Denton, E., Gross, S., and Fergus, R. **Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks.** 2016 <https://arxiv.org/abs/1611.06430>

Sencar, H. T., and Memon, N.

Digital Image Forensics: There is More to a Picture than Meets the Eye. ISBN-13: 978-1461407560

https://www.amazon.com/Digital-Image-Forensics-There-Picture/dp/1461407567/ref=reg_hu_rd_add_1_dp

The National Academy of Science. Strengthening Forensic Science in the United States: A Path Forward. 2009. *The National Academies Press.* <https://www.ncjrs.gov/pdffiles1/nij/grants/228091.pdf>

Esther Shein is a freelance technology and business writer based in the Boston area.

DOI:10.1145/3180485

Ross Anderson

Privacy and Security Making Security Sustainable

Can there be an Internet of durable goods?

AS WE START to connect durable goods such as cars, medical devices, and electricity meters to the Internet, there will be at least three big changes. First, security will be more about safety than privacy. Certification will no longer mean testing a car once before selling it for 10 years; safety will mean monthly software updates, and security will be an integral part of it. Second, we will have to reorganize government functions such as safety regulators, standards bodies, testing labs, and law enforcement. Finally, while you might get security upgrades for your phone for two or three years, cars will need safety and security patches for 20 years or more. We have no idea how to patch 20-year-old software; so we will need fresh thinking about compilers, verification, testing, and much else.

Privacy, Availability, or Safety?

The early security scares about the “Internet of Things” have mostly been about privacy. There have been reports of the CIA and GCHQ turning smart TVs into room bugs, while the

German government banned the Cayla doll whose voice-recognition system could be abused in the same way.³ Yet privacy may change less than we think. Your car knows your location history, sure, but your phone knows that already. It also knows where you walk, and it is already full of adware.

Denial of service has also been in the news. In October 2016, the Mirai botnet used 200,000 CCTV cameras (many of them in Brazil and Vietnam) to knock out Twitter in the Eastern U.S. for several hours. ISPs know they may have to deal with large floods of traffic from senders with whom they cannot negotiate, and are starting to get worried about the cost.

But the most important issue in the future is likely to be safety. Phones and laptops do not kill a lot of people, at least directly; cars and medical devices do.

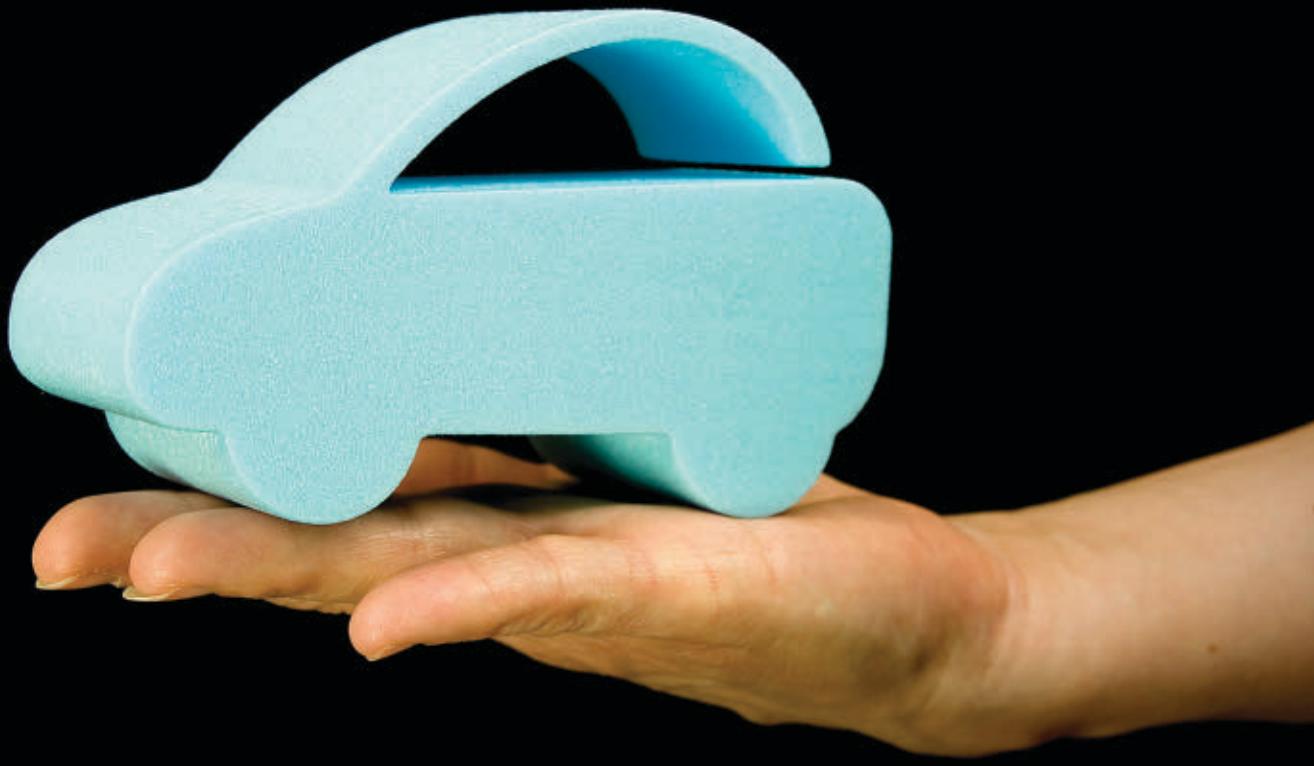
In 2016, Éireann Leverett, Richard Clayton, and I conducted a research project for the European Commission on what happens when devices that are subject to safety regulation start to contain both computers and communications.⁵ There are surprisingly

many regulated industries; it is not just the obvious ones like aircraft and railway signals, but even kids’ toys—they must not have lead paint, and if you pull a teddy bear’s arms off, they must not leave sharp spikes.

So what is the strategic approach? We looked at three verticals—road vehicles, medical devices, and smart meters. Cars are a good example to illustrate what we learned—though the lessons apply elsewhere too.

Security and Safety for Cars

Car safety has been regulated for years. In the U.S., the National Highways Transportation and Safety Administration was established in the 1960s following Ralph Nader’s campaigning; Europe has an even more complex regulatory ecosystem. Regulators discovered by the 1970s that simply doing crash tests and publishing safety data were not enough to change industry behavior. They had to set standards for type approval, mandate recalls when needed, and coordinate car safety with road design and driver training. Insurers do some of the regulatory work, as



do industry bodies; but governments provide the foundation.

This ecosystem faces a big change of tempo. At present, a carmaker builds a few hundred prototypes, sends some for crash testing, has the software inspected, and gets certification under more than 100 regulations. Then it ramps up production and sells millions of cars. Occasionally carmakers have to change things quickly. When a Swedish journalist found that avoiding an elk could cause an A-class car model to roll over, Mercedes had to redesign its suspension and fit a stability control system, which delayed the product launch at a cost of \$200 million.² But most of the safety case is a large up-front capital cost, while the time constant for the design, approval, and testing cycle is five years or so.

In the future, a vulnerability in a car will not need a skillful automotive journalist to exploit it. Malware can do that. So if a car can be crashed by commands issued remotely over the Internet, it will have to be fixed. Although we have known for years that car software could be hacked, the first widely

publicized public demonstration that a Jeep Cherokee could actually be run off the road—by Charlie Miller and Chris Valasek, in 2015—showed that the public will not tolerate the risk.⁴ While previous academic papers on car hacking had been greeted with a shrug, press photos of the Cherokee in a ditch forced Chrysler to recall 1.4 million vehicles.

Cars, like phones and laptops, will get monthly software upgrades. Tesla

has started over-the-air upgrades, and other vendors are following suit. This will move safety regulation from pre-market testing to a safety case maintained in real time—a challenge for both makers and regulators. We will need better, faster, and more transparent reporting of safety and security incidents. We will need responsible disclosure—so people who report problems are not afraid of lawsuits. We will need to shake up existing regulators, test labs, and standards bodies. Over two dozen European agencies have a role in car safety, and none of them have any cybersecurity expertise yet. We will need to figure out where the security engineers are going to sit.

We may need to revisit the argument between intelligence agencies who want “exceptional access” to systems for surveillance, and security experts who warn that this is hazardous.¹ The Director of the FBI and the U.K. Home Secretary both argue that they should be able to defeat encryption; they want a golden master key to your phone so they can listen in. But how many would agree to a golden master

Over two dozen European agencies have a role in car safety, and none of them have any cybersecurity expertise yet.

key that would let government agents crash their car?

There are opportunities too. Your monthly upgrade to your car software will not just fix the latest format string vulnerability, but safety flaws as well. The move to self-driving cars will lead to rapid innovation with real safety consequences. At present, product recalls cost billions, and manufacturers fight hard to avoid them; in the future, software patches will provide a much cheaper recall mechanism, so we can remove the causes of many accidents with software, just as we now fix dangerous road junctions physically.

But cars will still be more difficult to upgrade than phones. A modern car has dozens of processors, in everything from engine control and navigation through the entertainment system to the seats, side mirrors, and tire-pressure sensors. The manufacturer will have to coordinate and drive the process of updating subsystems and liaising with all the different suppliers. Its “lab car”—the rig that lets test engineers make sure everything works together—is already complex and expensive, and the process is about to get more complex still.

Sustainable Safety and Security

Perhaps the biggest challenge will be durability. At present most vendors won’t even patch a three-year-old phone. Yet the average age of a U.K. car at scrappage is 14.8 years, and rising all the time; cars used to last 100,000 miles in the 1980s but now keep going for nearer 200,000. As the embedded carbon cost of a car is about equal to that of the fuel it will burn over its lifetime, a significant reduction in vehicle durability will be unacceptable on environmental grounds.

As we build more complex artifacts, which last longer and are more safety critical, the long-term maintenance cost may become the limiting factor. Two things follow. First, software sustainability will be a big research challenge for computer scientists. Second, it will also be a major business opportunity for firms who can cut the cost.

On the technical side, at present it is hard to patch even five-year-old software. The toolchain usually will not compile on a modern platform,

Perhaps the biggest challenge will be durability.

leaving options such as keeping the original development environment of computers and test rigs, but not connecting it to the Internet. Could we develop on virtual platforms that would support multiple versions?

That can be more difficult than it initially appears. Toolchain upgrades already break perfectly functional software. A bugbear of security developers is that new compilers may realize that the instructions you inserted to make cryptographic algorithms execute in constant time, or to zeroise cryptographic keys, do not affect the output. So they optimize them away, leaving your code suddenly open to side-channel attacks. (In separate work, Laurent Simon, David Chisnall, and I have worked on compiler annotations that enable a security developer’s intent to be made explicit.)

Carmakers currently think their liability for upgrades ends five years after the last car is sold. But their legal obligation to provide spare parts lasts for 10 years in Europe; and most of the cars in Africa arrive in the country secondhand, and are repaired for as long as possible to keep them operable. Once security patches become necessary for safety, who is going to be writing the patches for today’s cars in Africa in 25 years’ time?

This brings us to the business side—to the question of who will pay for it all. Markets will provide part of the answer; insurance premiums are now rising because low-speed impacts now damage cameras, lidars, and ultrasonic sensors, so that a damaged side mirror can cost \$1,000 rather than \$100. The firms that earn money from these components have an incentive to help maintain the software that uses them. And part of the answer

will be legal; there have been regulations in Europe since 2010 that force carmakers to provide technical information to independent garages and spare-parts manufacturers. It is tempting to hope that a free/open source approach might do some of the heavy lifting, but many critical components are proprietary, and need specialist test equipment for software development. We also need incentives for minimalism rather than toolchain bloat. We do not really know how to allocate long-term ownership costs between the different stakeholders so as to get the socially optimal outcome, and we can expect some serious policy arguments. But whoever pays for it, dangerous bugs have to be fixed.

Once software becomes pervasive in devices that surround us, that are online, and that can kill us, the software industry will have to come of age. As security becomes ever more about safety rather than just privacy, we will have sharper policy debates about surveillance, competition, and consumer protection. The notion that software engineers are not responsible for things that go wrong will be put to rest for good, and we will have to work out how to develop and maintain code that will go on working dependably for decades in environments that change and evolve. □

References

- Abelson, H. et al. Keys under doormats: Mandating insecurity by requiring government access to all data and communications. *Commun. ACM* 58, 10 (Oct. 2015).
- Andrews, E.L. Mercedes-Benz tries to put a persistent Moore problem to rest. *New York Times* (Dec. 11, 1997).
- German parents told to destroy Cayla dolls over hacking fears. *BBC News* (Feb. 17, 2017).
- Greenberg, A. Hackers remotely kill a jeep on the highway—with me in it. *Wired* (July 21, 2015).
- Leverett, É., Clayton, R., and Anderson, R. Standardisation and certification of the Internet of Things. In *Proceedings of WEIS 2017*; <http://weis2017.econinfosec.org/program/>.

Ross Anderson (Ross.Anderson@cl.cam.ac.uk) is Professor of Security Engineering at Cambridge University, U.K. He is a Fellow of the Royal Society and the Royal Academy of Engineering, and author of *Security Engineering—A Guide to Building Dependable Distributed Systems*.



DOI:10.1145/3180488

Pamela Samuelson

Legally Speaking

Will the Supreme Court Nix Reviews of Bad Patents?

Considering the longer-term implications of a soon-to-be-decided U.S. Supreme Court case.

BAD" PATENTS HAVE been a plague to many in the software industry. Patents can be "bad" for numerous reasons. Although patents are supposed to be available only for new and inventive advances, sometimes it is difficult for examiners to locate the most relevant prior art. Not knowing about this art may cause them to approve patents that should not have been issued. Sometimes claims are too abstract or vague to be eligible for patents, or are deficient in other ways.

To address the bad patent problem, most developed countries have created administrative procedures so that third parties can challenge the validity of patents by asking a patent office tribunal to re-examine the patents. Post-grant review procedures are cost-effective ways to get rid of bad patents without having to go through full-dress, multiyear, very expensive litigation and appellate review. Patents that survive post-grant reviews are "stronger" for having gone through this extra scrutiny.

In 2011, the U.S. Congress enacted the America Invents Act (AIA), which, among other things, gave the Patent Trial & Appeal Board (PTAB) the power to review issued patents and extinguish claims that the board deems deficient for lack of novelty or inventiveness. Those dissatisfied with



The statue *The Contemplation of Justice*, created by the sculptor James Earle Fraser, flanks the U.S. Supreme Court building in Washington, D.C.

PTAB rulings can ask the Court of Appeals for the Federal Circuit (CAFC) to review them.

Greene's Energy Group was among the firms that asked PTAB to review a patent being asserted against it. Oil States Energy Services had sued it in federal court for infringement of this patent. PTAB agreed with Greene's that the Oil States patent was invalid on

lack of novelty grounds. Oil States appealed this decision to the CAFC, in part by attacking the constitutionality of the PTAB review process.

The CAFC thought so little of Oil States' appeal that it did not even issue an opinion to explain its denial. But Oil States was not ready to give up. It asked the U.S. Supreme Court to review the constitutional question.

ACM Transactions on Accessible Computing



ACM TACCESS is a quarterly journal that publishes refereed articles addressing issues of computing as it impacts the lives of people with disabilities. The journal will be of particular interest to SIGACCESS members and delegates to its affiliated conference (i.e., ASSETS), as well as other international accessibility conferences.



For further information or to submit your manuscript, visit taccess.acm.org

To everyone's astonishment, the Court decided in June 2017 to hear *Oil States*' appeal. All of a sudden, the very useful PTAB tool that Congress created to enable challenges to "bad" patents was at risk of being struck down on hoary old constitutional grounds.

What's at Stake?

Oil States is obviously upset at the invalidation of its patent on a device and method to cap well-heads during hydraulic fracturing (aka fracking) procedures so the well-heads can withstand considerable pressure caused when firms pump fluids into oil and gas wells to stimulate or increase the production.

The *Oil States* lawsuit against Greene's for infringing this patent was stayed during the PTAB review. Unless the Supreme Court invalidates the whole PTAB review procedure on constitutional grounds, *Oil States* will lose this lawsuit.

The stakes are, of course, much larger than the loss that *Oil States* would sustain if the Supreme Court rules against it. In its petition asking the Court to hear its appeal, *Oil States* emphasized that PTAB had overturned nearly 80% of the patent claims it had reviewed. (This affected almost 10,000 patents as of March 2016.) That may sound like a lot, but consider the PTAB only reviews patents when it has decided the challenges are likely to succeed.) The *Oil States* brief cited one source estimating that PTAB reviews had "destroyed" \$546 billion in value to the U.S. economy and "wiped out" \$1 trillion in valuation of companies whose patents had been invalidated.

If PTAB was right that these patents should not have issued, one might think it is good that companies sued for infringement did not have to pay \$546 billion to license invalid patents. The \$1 trillion in lower valuation for firms whose patents were struck down may also be well deserved. That, however, is not directly relevant to the constitutional issue the Court will be grappling with in the *Oil States* case.

Relevant Constitutional Provisions

Explaining the legal issue before the Supreme Court in *Oil States* requires a brief review of key parts of the U.S. Constitution. Article I confers numerous legislative powers on the U.S. Con-

gress, including the power to pass laws that grant exclusive rights for limited times to inventors for their discoveries in the useful arts (that is, patents).

Article II establishes the Executive Branch of the U.S. government and sets forth powers of the President and those who work within that branch of the government.

Article III establishes and gives certain powers to federal courts. Section 1 states: "The judicial Power of the United States, shall be vested in one supreme Court, and in such inferior Courts as the Congress may from time to time ordain and establish." Section 2 provides in relevant part that "[t]he judicial Power shall extend to all Cases, in Law and Equity, arising under this Constitution, the Laws of the United States, and Treaties made, or which shall be made, under their Authority."

Also at issue in *Oil States* is the Seventh Amendment to the Constitution, one of the 10 amendments to the Constitution (widely known as the Bill of Rights), which were promulgated during the process of state ratification of the Constitution to address several civil liberties concerns raised in debates about the original document.

The Seventh Amendment provides: "In Suits at common law, where the value in controversy shall exceed twenty dollars, the right of trial by jury shall be preserved, and no fact tried by a jury, shall be otherwise re-examined in any Court of the United States, than according to the rules of the common law."

Oil States' Constitutional Attack

Oil States' argument, boiled down to its essence, is this: Congress created

Virtually everyone who pays attention to patent law was surprised by the Court's decision to review the CAFC's decision in *Oil States*.

the PTAB as an administrative tribunal within of the U.S. Patent and Trademark Office (PTO). Congress gave PTAB the power to adjudicate certain challenges to the validity of issued patent claims. Under its theory, Oil States contends that only Article III courts have the constitutional authority to adjudicate patent validity. It further claims that under the Seventh Amendment, it has the right to a jury trial on its patent claims.

Oil States relies on an 1898 Supreme Court opinion, *McCormick Harvesting Machine Co. v. C. Aultman & Co.*, which says, among other things, that a patent "is not subject to being revoked or cancelled by the President, or any other officer of the government" because "it has become the property of the patentee, and as such is entitled to the same legal protection as other property." Because courts had, prior to adoption of the Constitution, the power to review patent validity and infringement, the Constitution should be understood to have vested Article III courts, and only Article III courts, with the power to decide whether patents are valid.

Oil States is not the first firm to have raised this particular constitutional challenge. Three times previously the Supreme Court denied petitions to review the constitutionality of the PTAB power to extinguish patent claims. This explains why virtually everyone who pays attention to patent law was surprised by Court's decision to review the CAFC's decision in *Oil States*.

The *Oil States* case has many owners of weak patents hoping the Court will find merit in Oil States' arguments. Many of others are worried that the Court will get tangled up in abstract arguments about whether patents are "private" or "public" rights under the Court's complicated jurisprudence about powers of Article III courts and powers that Congress has to create administrative tribunals to handle certain kinds of claims.

Responses to Oil States' Claims

Greene's had hoped to ward off Supreme Court review by relying on the Court's prior lack of interest in hearing this kind of challenge. On the merits, it relied upon a 1989 Supreme Court opinion, *Gianfinaciera*

If the only way a bad patent can get killed is through a \$5–\$10 million federal court lawsuit, lots of bad patents will go unchallenged.

SA v. Nordberg, which opined that Congress has the power to decline to provide for jury trials to resolve certain types of disputes between private parties over statutory rights as long as the rights at issue are integral to a public regulatory scheme whose operations Congress has assigned to an administrative agency.

Greene's also cited precedents holding that patents are "public rights" and integrally related to the complex regulatory scheme that Congress assigned to the PTO. Because PTAB's reviews are part and parcel of that scheme, Greene's argued that Congress had power to assign these reviews to the expert agency in charge of this scheme, namely, the PTO.

The PTO also filed a brief in response to Oil States' petition, pointing out that Congress intended for the AIA to "establish a more efficient and streamlined patent system that will improve patent quality and limit unnecessary and counterproductive litigation" in response to a "growing sense that questionable patents are too easily obtained and are too difficult to challenge." The PTO brief argued that the Court should not thwart this laudable goal.

It noted that Congress had given several agencies besides the PTO the power to review and correct errors in their past decisions. PTO examiners, like many other federal agents, sometimes make mistakes. The PTAB review process enables correction of those mistakes.

The PTO challenged Oil States' reading of the *McCormick* decision. The holding in that case was that the patent

office could not cancel a patent because Congress had not given the office power to do so. By enacting the AIA, however, Congress conferred such power on the PTO. The sentences from *McCormick* on which Oil States relied were only "dicta."

Conclusion

Constitutional challenges to Congressional enactments are rarely successful. The Supreme Court generally gives a broad reading to the constitutional powers that Congress has to accomplish its goals, including the establishment of administrative tribunals. Even so, there is reason for PTAB-proponents to be worried about the *Oil States* case. The Supreme Court's jurisprudence on the powers of Article III courts versus the powers of Congress to establish tribunals to resolve private disputes is very arcane, convoluted, and far from consistent.

The Court's job is to interpret the Constitution. So the Justices cannot just decide that Congress had a good reason to establish PTAB and give it power to extinguish erroneously granted patents. The Justices will have to carve a careful path through the thicket of their past decisions to articulate standards to uphold PTAB's patent reviews, and give guidance about Congress' powers to establish other administrative tribunals, such as a small claims tribunal within the Copyright Office.

One can hope the Justices will consider the constitutional issue in *Oil States* in light of another constitutional purpose, that which underlies the patent system: promoting the progress of useful arts. If the only way a bad patent can get killed is through a \$5 million–\$10 million federal court lawsuit, lots of bad patents will go unchallenged. PTAB is an efficient mechanism to extinguish erroneously issued patents. It will be unfortunate indeed if the Court feels so caught in the web of its constitutional precedents that it cannot find a way to uphold the good work the PTAB has been doing. □

Pamela Samuelson (pam@law.berkeley.edu) is the Richard M. Sherman Distinguished Professor of Law and Information at the University of California, Berkeley, USA, and a member of the ACM Council.

Copyright held by author.

Computing Ethics Ethics Omission Increases Gases Emission

A look in the rearview mirror at Volkswagen software engineering.

THE VOLKSWAGEN EMISSIONS scandal came to light in September 2015. The company installed software into millions of vehicles with diesel engines so that impressive emission readings would be recorded in laboratory conditions even though the reality is that the diesel engines do not comply with current emission regulations. Volkswagen is a global organization headquartered in Germany; its subsidiaries adhere to common policies and a corporate culture. This worldwide scandal broke first in the U.S. with ongoing investigation and legal action there and in other countries including Germany, Italy, and the U.K.

Combustion engines are the source of pollution and therefore have been subjected to emission control. The formation of NOx (nitrogen oxides) through combustion is a significant contributor to ground-level ozone and fine particle pollution that is a health risk. On this basis, the use of software to control emissions must be defined as safety critical for, if it fails or malfunctions, it can cause death or serious injury to people. There does not appear to be any acknowledgement of this across vehicle manufacturing.

The statement from the U.S. Department of Justice¹⁶ details the facts of the VW emissions case. Two senior managers, Jens Hadler and Richard Dorenkamp appear to be at the center of the so-called defeat software's ongoing design and implementation processes. These began in 2006, with



A wall of rear-view mirrors at the entrance to Seat Pavilion in Volkswagen Group's Autostadt ("Car City") visitor attraction adjacent to the Volkswagen factory in Wolfsburg, Germany.

the design of a new diesel engine to meet stricter U.S. emission standards to take effect in 2007. The goal was to market new vehicles as meeting the stricter standards and attract U.S. buyers. Being unable to accomplish this, the engineers working under Hadler and Dorenkamp, developed software that allowed vehicles to distinguish test mode from drive mode thus satisfying the emissions test while allowing much greater emissions when vehicles were on the road. "Hadler authorized Dorenkamp to proceed with the project knowing that only the

use of the defeat device software would enable VW diesel vehicles to pass U.S. emissions tests."

Drawing upon the *Statement of Facts*, Leggett³ reported that although there had been some concerns over the propriety of the defeat software all those involved in the discussions including engineers were instructed not to get caught and furthermore to destroy related documents. According to Mansouri,⁴ Volkswagen is an autocratic company with a reputation for avoiding dissent and discussion. It has a compliant business culture where em-

ployees are aware that underperformance can result in replacement and so management demands must be met to ensure job security. Three statements in particular in the *Volkswagen Group Code of Conduct: Promotion of Interests, Secrecy, and Responsibility for Compliance*,¹⁷ align with the ongoing conduct encouraged during the emissions debacle. Trope and Ressler¹⁵ explain that as an autocratic book of rules, the group code supports and even promotes dishonest dysfunctional behavior that includes the creation of software to cheat, rather than solve, engineering problems and to protect that software from disclosure as if it were a trade secret.

In January 2017, the U.S. Justice Department announced that, "Volkswagen had agreed to plead guilty to three criminal felony counts, and pay a \$2.8 billion criminal penalty, as a result of the company's long-running scheme to sell approximately 590,000 diesel vehicles in the U.S. by using a defeat device to cheat on emissions tests mandated by the Environmental Protection Agency (EPA) and the California Air Resources Board (CARB), and lying and obstructing justice to further the scheme."

Business Analysis

Many of the accounts about the Volkswagen emissions case focus on business ethics with only a few touching upon the role of the software engineers in this situation. These accounts at times are repetitive but intertwine to provide a rich view. The widespread unethical actions across Volkswagen can be described as a new type of irresponsible behavior, namely *deceptive manipulation*.¹² The detail of this and the associated corporate repercussions are discussed further by Stanwick and Stanwick.¹⁴

Software engineers at Volkswagen faced ethical and legal issues that are easy to identify. Plant⁶ suggests that they should have alerted external bodies since the internal lines of reporting were compromised. Merkel⁵ concurs, citing the *Software Engineering Code of Ethics and Professional Practice* (see <http://www.acm.org/about/se-code>) by way of justification, and adds that the lack of whistleblowers in such a large group is surprising. Both authors point to the potential personal cost of whis-

Software engineers at Volkswagen faced ethical and legal issues that are easy to identify.

tleblowing as the reason it did not happen. Rhodes⁹ adds a second factor, arguing that corporate business ethics is very much a *pro-business stance* that is implemented through corporate control and compliance systems, and instruments of managerial coordination. This can enable the pursuit of business self-interest through organized widespread conspiracies involving lying, cheating, fraud, and lawlessness. This is what happened at Volkswagen. Queen⁷ concurs, explaining that Volkswagen intentionally deceived those to whom it owed a duty of honesty. The pressure for continuous growth and the perception that failure was not an option⁸ created a culture where corporate secrecy was paramount—which in turn implicitly outlawed whistleblowing.

The Role of Software Engineering

If one has a responsibility for the planning, design, programming, or implementation of software then that aspect of one's work falls within the scope of the *Software Engineering Code of Ethics and Professional Practice* regardless of one's job title. In that sense software engineering pervades this debacle and is therefore worthy of further investigation.

So what was the role of software engineers in the creation and installation of VW's defeat software? This question can be addressed using the *Software Engineering Code of Ethics and Professional Practice*. The code is long established, documenting the ethical and professional obligations of software engineers and identifying the standards society expects of them.² The code translates ethical principles into practical guidance. It encourages positive action and resistance to act unethically. It has been adopted by many professional bodies and companies worldwide and

has been translated into Arabic, Croatian, French, German, Hebrew, Italian, Mandarin, Japanese, and Spanish (see <http://bit.ly/2nlOYro>).

Software engineers and software engineering educators have a responsibility to be cognizant of the code and its requirements. *Public Interest*, which is apposite for safety-critical software, is central to the code. Although education can influence the courage and capability to act in accordance with the code, that result depends on structural and psychological supports within the environment in which engineers practice.

The actions of VW managers and software engineers violated the following principles of the code:

Principle 1.03 "approve software only if they have a well-founded belief that it is safe, meets specifications, passes appropriate tests, and does not diminish quality of life, diminish privacy, or harm the environment. The ultimate effect of the work should be to the public good." The defeat software is clearly unsafe given NOx pollution damages both health and the environment. The public were under the misapprehension that VW cars were emitting low levels of NOx and therefore not a health risk. Thus, software engineers installed unethical software.

Principle 1.04 "disclose to appropriate persons or authorities any actual or potential danger to the user, the public, or the environment, that they reasonably believe to be associated with software or related documents." There is no evidence that any software engineer disclosed. Commercial software is usually developed in teams and in this case it is likely this was a large team spanning all aspect of software development.

Principle 1.06 "be fair and avoid deception in all statements, particularly public ones, concerning software or related documents, methods and tools." The emissions software was heralded publicly as a success when internally there was widespread knowledge that this claim was fraudulent. Software engineers were likely to have been privy to this cover-up.

Principle 2.07 "identify, document, and report significant issues of social concern, of which they are aware, in software or related docu-

ments, to the employer or the client." There is some evidence that concern was raised about the efficacy of the defeat software but it seems those in dissent allowed themselves to be managed towards deception.

Principle 3.03 "identify, define and address ethical, economic, cultural, legal and environmental issues related to work projects." The EPA regulations are explicit and are legally binding. From the evidence accessed it is unclear as to whether software engineers knew of the illegality of their actions. Nevertheless ignorance cannot and must not be a form of defence.

Principle 6.06 "obey all laws governing [the] work, unless, in exceptional circumstances, such compliance is inconsistent with the public interest." This relates to the analysis under principle 3.03. Compliance to further the prosperity of Volkswagen was at the expense of legal compliance.

Principle 6.07 "be accurate in stating the characteristics of software on which they work, avoiding not only false claims but also claims that might reasonably be supposed to be speculative, vacuous, deceptive, misleading, or doubtful." Software engineers could argue internally that the software indeed performed as it was designed to. However, the design was to achieve regulatory and public deception.

Principle 6.13 "report significant violations of this Code to appropriate authorities when it is clear that consultation with people involved in these significant violations is impossible, counterproductive or dangerous." Given the apparent corporate culture within Volkswagen there was little point in reporting concerns further up the line. In fact the corporate code seems at odds with the professional code regarding this point. Software engineers failed to report these breaches to appropriate authorities.

Conclusion

Professionals, who must have been party to this illegal and unethical act, developed and implemented this software. Those who undertake the planning, development, and operation of software have obligations to ensure integrity of output and overall to contribute to the public good.¹⁰ The ethical practice of software engineers is

Unethical actions related to software engineering can be addressed from two sides.

paramount. Practice comprises process and product. Process concerns virtuous conduct of software engineers, whereas product concerns whether software is deemed to be ethically viable. Actions and outcomes in the Volkswagen case appear to have failed on both counts.

These serious issues related to professional practice must be addressed. It is hoped such issues are exceptional but sadly it is likely they are commonplace given the ongoing plethora of software disasters (see, for example, Catalogue of Catastrophe¹ and Software Fail Watch¹³). Unethical actions related to software engineering can be addressed from two sides. One side focuses on resisting the temptation to perform unethical practice while the other focuses on reducing the opportunity of performing unethical practice. Society at large needs competent, ethical, and altruistic professionals to deliver societally acceptable, fit-for-purpose software. Both of these can be helped by education, but education will not suffice without adequate social supports.

In order to fulfill software engineering duties, an individual must fully understand the professional responsibilities and obligations of the role. These are explicitly laid out in the *Software Engineering Code of Ethics and Professional Practice* and as such individuals must know and apply it to their everyday work. To achieve this, the effective education of new professionals is essential. Teaching technology in isolation is unacceptable and dangerous. Software engineers need a broader education to gain the necessary skills and knowledge to act in a socially responsible manner not on the basis of instinct and anecdote but

on rigor and justification.¹¹ They must possess practical skills to address the complex ethical and societal issues that surround evolving and emerging technology. Such education should be based on a varied diet of participative experiential learning delivered by those who have a practical understanding of the design, development, and delivery of software. Contrasting the Volkswagen Group Code of Conduct with the Software Engineering Code might provide one means for experiential learning. Such educated software engineers might find ways to prevent the installation of unethical software of the future. C

References

- Catalogue of Catastrophe, International Project Leadership Academy; <http://bit.ly/2FvCtkl>.
- Gotterbarn, D., Miller, K. and Rogerson, S. Software engineering code of ethics is approved. *Commun. ACM* 42, 10 (Oct. 1999), 102–107 and *Computer* (Oct. 1999), 84–89; <http://bit.ly/2nI0Yro>.
- Leggett, T. VW papers shed light on emissions scandal. BBC News (Jan. 12, 2017); <http://bbc.in/2AWMtzW>.
- Mansouri, N. A case study of Volkswagen unethical practice in diesel emission test. *International Journal of Science and Engineering Applications* 5, 4 (2016), 211–216.
- Merkel, R. Where were the whistleblowers in the Volkswagen emissions scandal? *The Conversation* 30, (Sept. 30, 2015); <http://bit.ly/2AUeGac>.
- Plant, R. A software engineer reflects on the VW scandal. *The Wall Street Journal* (Oct. 15, 2015); <http://on.wsj.com/2FsKruB>.
- Queen, E.L. How could VW be so dumb? Blame the unethical culture endemic in business. *The Conversation* (Sept. 26, 2015); <http://bit.ly/1LZV0jO>.
- Ragatz, J.A. What can we learn from the Volkswagen scandal? Faculty Publications, 2015, Paper 297; <http://bit.ly/2CW8XTr>.
- Rhodes, C. Democratic business ethics: Volkswagen's emissions scandal and the disruption of corporate sovereignty. *Organization Studies* 37, 10 (2016), 1501–1518.
- Rogerson, S. Ethics and ICT. In R. Galliers and W. Currie, Eds. *The Oxford Handbook on Management Information Systems: Critical Perspectives and New Directions*. Oxford University Press, 2011, 601–622.
- Rogerson, S. Future Vision, Special Issue—20 years of ETHICOMP. *Journal of Information, Communication and Ethics in Society* 13, 3/4 (2015), 348–360.
- Siano, A. et al. More than words: Expanding the taxonomy of greenwashing after the Volkswagen scandal. *Journal of Business Research* 71(C), (2017), 27–37.
- Software Fail Watch 2016, Quarter One, Tricentis; <http://bit.ly/2r2nnzy>.
- Stanwick, P. and Stanwick, S. Volkswagen emissions scandal: The perils of installing illegal software. *International Review of Management and Business Research* 6, 1 (2017), 18.
- Trope, R.L. and Ressler, E.K. Mettle fatigue: VW's single-point-of-failure ethics. *IEEE Security & Privacy* 14, 1 (2016), 12–30.
- U.S. Department of Justice. Volkswagen AG agrees to plead guilty and pay \$4.3 billion in criminal and civil penalties and six Volkswagen executives and employees are indicted in connection with conspiracy to cheat U.S. emissions tests. *Justice News*, (Jan. 11, 2017); <http://bit.ly/2j7mT5M>.
- Volkswagen. The Volkswagen Group Code of Conduct, 2010; <http://bit.ly/2IIYPF8>.

Simon Rogerson (srog@dmu.ac.uk) is Professor Emeritus of Computer Ethics in the Centre for Computing and Social Responsibility at De Montfort University, The Gateway, Leicester, U.K.

Copyright held by author.



DOI:10.1145/3182108

Peter J. Denning

The Profession of IT The Computing Profession

*Taking stock of progress toward a computing profession
since this column started in 2001.*

WE STARTED THIS column in 2001 when ACM was re-envisioning itself as a society of a computing profession. ACM leaders and many members already thought of computing as a profession. They wanted ACM to strengthen its support of computing professionals and its commitment to the practitioners of a computing profession. How has all this progressed in the past 17 years?

In my first column on the IT profession, my opening question was whether a profession is needed in the first place. I wrote: “To most of the hundreds of millions of computer users around the world, the inner workings of a computer are an utter mystery. Opening the box holds as much attraction as lifting the hood of a car. Users look to computing professionals to help them with their needs for designing, locating, retrieving, using, configuring, programming, maintaining and understanding computers, networks, applications, and digital objects.”¹ The need has intensified over the years because there are now billions of users and the technologies they rely on are much more complex.

The ACM and IEEE Computer Society are the two main professional societies in computing. They are comparable in size with approximately 100,000 members each. ACM has traditionally emphasized the science-math side of computing, and IEEE-CS the engineering-design side of computing. The two societies have cooperated on many



joint ventures including curriculum recommendations and accreditation. They have diverged on certification and licensing, which traditionally have been eschewed by ACM leadership and embraced by IEEE-CS leadership.

The next question was what specialties have professionals organized to deal with specific kinds of concerns—for example, specialists in programming languages, operating systems, networks, or graphics. The ACM SIGs and IEEE-CS hosted organizations to support these groups. ACM had (and still has) around 40 SIGs in specialized areas. The list of SIGs is a useful guide to the organized core specialties of computing.

However, the list of ACM and IEEE-CS specialty organizations does not

come close to covering all the organized specialties in computing. There are two other categories—computing-intensive fields in science and engineering where computing is a tool but is not the focus of concern, and computing-infrastructure occupations, where specialists operate and maintain the infrastructures on which everyone depends. Table 1 is an update of the original table,^{1,2} now showing 52 specialties.

Table 1 reflects the interests of the members of ACM and IEEE-CS. However, this is not the only way to categorize computing professionals. The U.S. Bureau of Labor statistics maintains a list of “computer and information technology occupations” that spells out the kinds of jobs employers recruit

for. Table 2 summarizes the BLS information about computing.^a Table 1 can be seen as an enumeration of the specialties covered by the groups listed in Table 2. No matter how you look at it, there is a huge market for computing professionals.

Between them, the ACM, IEEE-CS, and a large network of education institutions provide an extensive support structure for computing professionals, including these elements:^{2,3}

- ▶ Curricula that grant entry into profession
- ▶ Standards for curricula (body of knowledge)
- ▶ Standards for professional practice
- ▶ Professional development (short courses, books)
- ▶ Accreditation guidelines and evaluation
- ▶ Certification
- ▶ Licensing
- ▶ Code of ethics
- ▶ Professional specialty groups

The professional societies do a lot of work to support computing professionals!

ACM and the Profession

The ACM's founders in 1947 believed that computers would be a permanent source of attention and concern, eventually permeating all fields. ACM's initial responses to this concern were a computing research journal (*Journal of the ACM*), a newsletter (*Communications of the ACM*), and a network of local chapters. Beginning in the early 1960s, ACM developed and maintained a code of ethics. The first computer science departments were founded in 1962 and ACM issued its first curriculum recommendations in 1968. Around 1985, ACM began partnering with IEEE-CS on accreditation and upgrades to curriculum recommendations. Over the years, ACM/IEEE-CS have evolved successively more sophisticated curriculum recommendations into a "computing body of knowledge." ACM has supported professionals in developing competencies through its professional development center and ongoing discussions about good professional practice in *Communications* and *ACM Queue* magazines.

^a <https://www.bls.gov/ooh/computer-and-information-technology/home.htm>

Table 1. Selected professional specialties of computing.

Computing-Core Disciplines	Computing-Intensive Fields	Computing-Infrastructure Occupations
Artificial intelligence	Aerospace engineering	Blockchain administrator
Cloud computing	Autonomous systems	Computer technician
Computer science	Bioinformatics	Data analyst
Computer engineering	Cognitive science	Data engineer
Computational science	Cryptography	Database administrator
Database engineering	Computational science	Help desk technician
Computer graphics	Data science	Identity theft recovery agent
Cyber security	Digital library science	Network technician
Human-computer interaction	E-commerce	Professional IT trainer
Network engineering	Genetic engineering	Reputation manager
Programming languages	Information science	Security specialist
Programming methods	Information systems	System administrator
Operating systems	Public Policy and Privacy	Web identity designer
Performance engineering	Instructional design	Web programmer
Robotics	Knowledge engineering	Web services designer
Scientific computing	Management information systems	
Software architecture	Network science	
Software engineering	Multimedia design	
	Telecommunications	

Table 2. BLS occupations.

Category	Entry Degree	Median Salary in 2016
Computer and Information Research Scientists	MS	\$112K
Software Developers	BS	\$102K
Computer Network Architects	BS	\$101K
Information Security Analysts	BS	\$93K
Computer Systems Analysts	BS	\$87K
Database Administrators	BS	\$85K
Computer Programmers (coders)	BS	\$80K
Network and Computer System Administrators	BS	\$80K
Web Developers	AS	\$66K
Computer Support Specialists	BS or AS	\$52K

Source: bls.gov

Since the 1960s, ACM has hosted more than 40 special interest groups (SIGs) in various specialties of computing.

In 2002, ACM established a Profession Board, later renamed the Practitioner Board, to establish and maintain programs for the professional development of the 70% of members who are practicing, nonacademic professionals. The Practitioner Board today offers an increasing range of programs to support professionals, including professional development, books, marketing, distinguished speakers, practitioner-oriented conferences, and GPAC (global practitioner advisory council), a global network of about 100 profes-

sionals who provide advice and ideas to the Board. The Practitioner Board partners with relevant ACM publications, for example in *Queue* and in *Ubiquity*'s advisory panel of young professionals.

Challenges

In 2001, I wrote that the biggest challenge for ACM in fostering a profession would be academic computer scientists giving up the illusion they could either control the profession or be seen as its leaders. Computing was spreading prolifically and professionals in other fields were independently organizing professional groups to support them. A conspicuous example was the compe-

tational sciences, such as computational physics, computational chemistry, or computational biology, where professional groups were being organized independent of ACM or IEEE. At the time, computer scientists were not very open to reaching out to other fields and helping them meet their own needs in computing. Over the years, ACM has embraced this challenge and has become much better at reaching out to other fields. ACM has settled into a role as a “curator of the flame,” providing the definitions, bodies of knowledge, and standards of practice for computing wherever it appears.

One of the consequences of the spread of computing into everyone’s lives is that the public mood has been shifting to the notion that essential knowledge to support education and advancement of the profession should be free to the public. The ACM Digital Library, which has been feeling the pressure of this mood for several years, now supports open access to research papers for which the authors have paid an up-front open access fee. The library itself is available to more practicing professionals because ACM grants access through licenses to organizations. These changes have reduced revenues for digital library and other publication subscriptions. While it does not have a final answer, ACM has been making good headway toward finding revenues to support its knowledge base in a way that it can ultimately be free to the public.

But ACM’s biggest challenge concerns the relations between two major sectors of its members. The “academic-research” sector is members who are on the faculty of universities and colleges or are employed by industry research labs. The “practitioner” sector is members who are practicing non-academic professionals, either self employed or employed by a company. ACM sometimes uses the term “industry professional” for practitioner.

One aspect of this challenge is bridging the gap between ACM’s treasure trove of research papers (in the digital library) and the working worlds of practitioners. Most research papers are communications among researchers that enable the advancement of a research field. Many practitioners, however, find these papers difficult at best and opaque at worst. Bridging the gap means find-

ing authors who understand both worlds and can translate the key ideas of research into useful ideas for practice. This is quite difficult because few such authors exist. A fine example is the The Morning Paper, a five-times weekly blog by Adrian Colyer in the U.K. (<http://blog.acolyer.org>); in each issue, Colyer translates a research paper into terms and connections that practitioners can use. Colyer has a relationship with ACM through the Practitioners Board.

Another aspect of this challenge is in leadership: ACM has lost its ability to populate its leadership positions with a mixture of academic-research and practitioner professionals so that these two worlds will get to know each other and work together for a stronger profession. All members of the ACM Council, and most of the SIG leadership, are from the academic-research sector. ACM has been particularly good at supporting its professional academic and research members with first-rate, widely respected publications, conferences, and awards. ACM has been less attentive to helping practitioners develop their professional skills, at articulating standards for essential professional skills, or at developing awards and other recognitions for industry professionals. The ACM Nominating Committee has its work cut out for it.

ACM could do much more in recognizing practitioner members for their contributions. Most ACM awards today go to members of the academic-research sector. The Distinguished Service Award, initially chartered in the 1960s as an industry professional award co-equal in prestige to the Turing Award, has faded into semi-obscenity and is now the only ACM award with no purse; it could be rejuvenated as a major recognition for senior practitioner members. ACM could also set up new awards explicitly for the practitioner sector.

Although ACM has major programs for practitioners—including professional development, learning center, and *Queue* magazine—practitioner members frequently tell us that ACM does not understand them. The Practitioner Board, under the leadership of Terry Coata and Stephen Ibaraki, has begun to turn this around, with 50 practitioner volunteers helping the board and another 100 providing advice through the GPAC network.

In my opinion, however, ACM will not achieve its goal of supporting a computing profession and its practitioners without a concerted effort to bring practitioners into ACM leadership positions and give them more professional recognitions. IEEE-CS has been more successful with this than ACM.

Self-Management

Individual members further their professional development by using the services and support structures of ACM and IEEE-CS, and by improving their own personal practices in their professional relations with clients. I have aimed the 64 The Profession of IT columns published since 2001 to support the latter. These columns have examined various aspects of the profession including the nature of the profession, education for professionals, innovation, language-action, the Internet, software, moods, jobs, and time management. You can find them on my website.^b Please contact me with your questions or issues that I can address in future columns.

Computing has come a long way in the 70 years since its founders’ first inklings that computing would become a pervasive professional concern. ACM has developed an impressive array of offers in publications, a digital library, conferences, chapters, support for professional education, support for practitioners, and awards. Its biggest challenge is integrating its academic-research and practitioner sectors. I expect significant progress on this challenge in the next decade. □

^b <http://denninginstitute.com/pjd/PUBS/CACMcols>

References

1. Denning, P.J. Who are we? *Commun. ACM* 44, 2 (Feb. 2001), 15–19.
2. Denning, P.J. and Frailey, D.J. Who are we—now? *Commun. ACM* 54, 6 (June 2011), 25–27.
3. Ford, G. and Gibbs, N. A Mature Profession of Software Engineering. Software Engineering Institute, Carnegie-Mellon University (1996); <http://www.sei.cmu.edu/library/abstracts/reports/96tr004.cfm>

Peter J. Denning (pjdenning@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for Information Innovation at the Naval Postgraduate School in Monterey, CA, USA. He is Editor of ACM Ubiquity, and is a past president of ACM. The author’s views expressed here are not necessarily those of his employer or the U.S. federal government.

Viewpoint

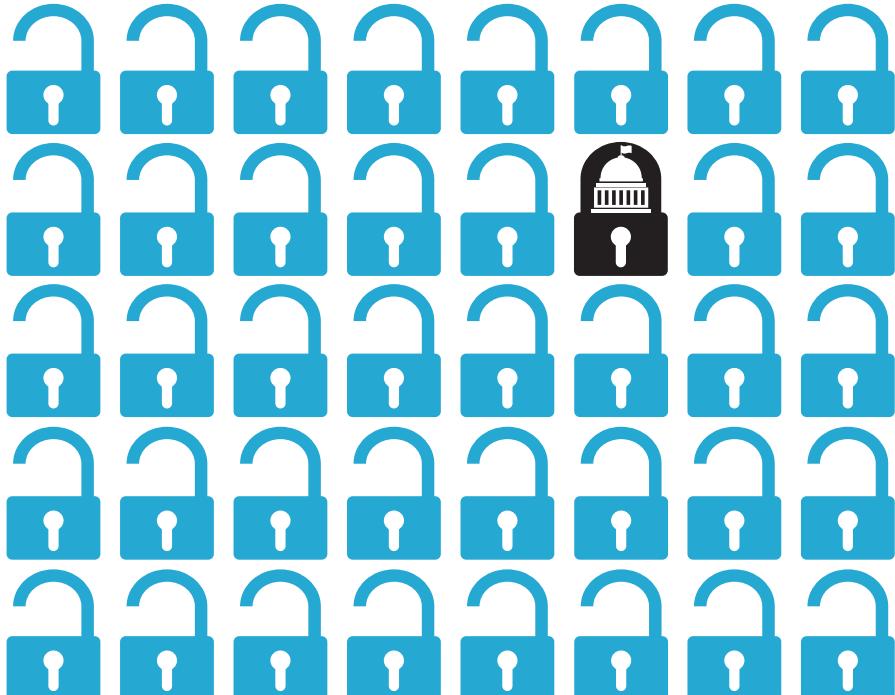
Impediments with Policy Interventions to Foster Cybersecurity

A call for discussion of governmental investment and intervention in support of cybersecurity.

THE LIST OF cyberattacks having significant impacts is long and getting longer, well known, and regularly invoked in calls for action. Such calls are not misplaced, because society is becoming more dependent on computing, making cyberattacks more capable of widespread harm. Vardi's recent call¹ "it is time to get government involved, via laws and regulations" motivates this Viewpoint. Indeed, we do know how to build more-secure systems than we are deploying today. And governments can—through regulation or other mechanisms—incentivize actions that individuals and organizations are otherwise unlikely to pursue.

However, a considerable distance must be traversed from declaring that government interventions are needed to deciding particulars for those interventions, much less intervening. To start, we need to agree on specific goals to be achieved. Such an agreement requires understanding monetary and other costs that we as a society are willing to incur, as well as understanding the level of threat to be thwarted. Only after such an agreement is reached, does it make sense for policymakers to contemplate implementation details.

This Viewpoint reviews interventions often suggested for incentivizing enhanced cybersecurity. I discuss



the trade-offs involved in the adoption of each. In so doing, I hope to facilitate discussions that will lead to agreements about goals and costs. It is premature to advocate for specific interventions, exactly because those discussions have yet to take place.

Secure Systems Are More Expensive

Assurance that a system will do what it should and will not do what it should not requires effort during develop-

ment. Somebody must pay. It could be consumers (through higher prices), government (through tax credits or grants), or investors (if developers will accept reduced profits). But realize that the consumers, taxpayers, and investors are just us. So before mandating expenditures for enhanced cybersecurity, we must decide that we are willing to pay and decide how much we are willing to pay.

Other priorities will compete. Some will advocate using "return on invest-

ment” (ROI) to set spending levels for cybersecurity versus other priorities. But ROI is problematic as a basis for justifying how much to spend here.

- There are no good ways to quantify how secure a system is. Measuring cybersecurity can be as difficult as establishing assurance for a system in the first place, which we know to be a hard problem for real systems.

- There are no good ways to quantify the costs of not investing in cybersecurity. To tally lost business or the work to recover data and systems ignores other, important harms from attacks. Disclosure of confidential information, for example, can destroy reputations, constrain future actions, or undermine advantages gained through technological superiority. Externalities also must be incorporated into a cost assessment—attacks can have both local and remote impact, because the utility of an individual computer often depends on, or is affected by, an entire network.

We should be mindful, though, that investments directed at other national priorities—defense, foreign aid, and social programs—are also difficult to evaluate in purely objective ways. Yet governments routinely prioritize across making such investments. Even in smaller, private-sector institutions, the “bottom line” is rarely all that matters, so they too have experience in making investment decisions when ROI or other objective measures are not available.

Any given intervention to encourage investing in cybersecurity will allocate costs across various sectors and, therefore, across different sets of individuals. A decision to invest in the first place might well depend on specifics of that allocation. We often strive to have those individuals who benefit the most be the ones who pay the most. But the nature of networked infrastructures makes it difficult to characterize who benefits from cybersecurity and by how much. For instance, civil government (and much of defense), private industry, and individuals all share the same networks and use the same software, so all benefit from the same security investments. Externalities also come into play. For example, should only the targeted political party be paying to prevent cyberattacks that,

The nature of networked infrastructures makes it difficult to characterize who benefits from cybersecurity.

if successful, threaten the integrity of an election outcome?

Investments in cybersecurity will have to be recurring. Software, like a new bridge or building, has both an initial construction cost and an ongoing maintenance cost. It is true that software does not wear out. Nevertheless, software must be maintained:

- Today’s approaches for establishing assurance in the systems we build have limitations. So some vulnerabilities are likely to remain in any system that gets deployed. When these vulnerabilities are discovered, patches must be developed and applied to systems that have been installed.

- Unanticipated uses and an environment that evolves by accretion mean that assumptions a system developer will have made might not remain valid forever. Such assumptions constitute vulnerabilities, creating further opportunities for attackers.

Ideally, systems will be structured to allow patching, and software producers will engage in the continuing effort to develop patches. Some business models (for example, licensing) are better than others (for example, sales) at creating the income stream needed to support that patch development.

Cost Is Not the Only Disincentive

Secure systems tend to be less convenient to use, because enforcement mechanisms often intrude on usability.

- One common approach for obstructing attacks is based on monitoring. The system authenticates each request before it is performed and uses the context of past actions when deciding what requests are authorized

to proceed. But user authentication requires (tedious) user interactions with the system; program authentication limits which software can be run on a system; and the role of context can limit a user’s flexibility in how tasks might be accomplished.

- Another common approach to defense is isolation. Here, effects of actions by users, programs, or machines are somehow contained. Isolation might be employed to keep attackers out or to keep attackers in. In either case, communications is blocked, which makes orchestrating cooperation difficult. We might, for example, facilitate secure access to a bank account by requiring use of a Web browser that is running in a separate (real or virtual) computer on which there is a separate file system and only certain “safe” application programs are available. The loss of access to other files or programs hinders attackers but it also hinders doing other tasks.

These enforcement mechanisms increase the chances that malicious actions will be prevented from executing, because they also block some actions that are not harmful. And users typically feel inconvenienced when limitations are imposed on how tasks must be accomplished. So nobody will be surprised to learn that users regularly disable enforcement mechanisms—security is secondary to efficiently getting the job done.

Security Can Be in Tension with Societal Values

Enhanced level of cybersecurity can conflict with societal values, such as privacy, openness, freedom of expression, opportunity to innovate, and access to information. Monitoring can undermine privacy; authentication of people can destroy anonymity; authentication of programs prevents change, which can interfere with flexibility in innovation and can be abused to block execution of software written by competitors. Such tensions must be resolved when designing interventions that will promote increased levels of cybersecurity.

Moreover, societal values differ across countries. We thus should not expect to formulate a single uniform set of cybersecurity goals that will serve for the entire Internet. In addition, the ju-

risdiction of any one government necessarily has a limited geographic scope. So government interventions designed to achieve goals in some geographic region (where that government has jurisdiction) must also accommodate the diversity in goals and enforcement mechanisms found in other regions.

Flawed Analogies Lead to Flawed Interventions

Long before there were computers, liability lawsuits served to incentivize the delivery of products and services that would perform as expected. Insurance was available to limit the insured's costs of (certain) harms, where the formulation and promulgation of standards facilitated decisions by insurers about eligibility for coverage. Finally, people and institutions were discouraged from malicious acts because their bad behavior would likely be detected and punished—deterrence.

Computers and software comprise a class of products and services, attackers are people and institutions. So it is tempting to expect that liability, insurance, and deterrence would suffice to incentivize investments to improve cybersecurity.

Liability. Rulings about liability for an artifact or service involve comparisons of observed performance with some understood basis for acceptable behaviors. That comparison is not possible today for software security, since software rarely comes with full specifications of what it should and should not do. Software developers and service providers shun providing detailed system specifications because specifications are expensive to create and could become an impediment to making changes to support deployment in new settings and to support new functionality. Having a single list that characterizes acceptable behavior for broad classes of systems (for example, operating systems or mail clients) also turns out to be problematic. First, by its nature, such a list could not rule out attacks to compromise a property that is specific only to some element in the class. Second, to the extent that such a list rules out repurposing functionality (and thereby blocks certain attacks), the list would limit opportunities for innovations (which often are imple-

Secure systems tend to be less convenient to use because enforcement mechanisms often intrude on usability.

mented by repurposing functionality).

Insurance. Insurance depends for pricing on the use of data about past incidents and payouts to predict future payouts. But there is no reason to believe that past attacks and compromises to computing systems are a good predictor of future attacks or compromises. I would hope successive versions of a given software component will be more robust, but that is not guaranteed. For example, new system versions often are developed to add features, and a version that adds features might well have more vulnerabilities than its predecessor. Moreover, software deployed in a large network is running in an environment that is likely to be changing. These changes—which might not be under the control of the developer, the user, the agent issuing insurance, or even any given national government—might facilitate attacks, and that further complicates the use of historical data for predicting future payouts.

Companies that offer insurance can benefit from requiring compliance with industrywide standards since the domain of eligible artifacts is now narrowed, which simplifies predictions about possible adverse incidents and payouts. Good security standards also will reduce the likelihood of adverse incidents. However, any security standard would be equivalent to a list of approved components or allowed classes of behavior. Such a list only can rule out certain attacks and it can limit opportunities for innovation, so security standards are unlikely to be popular with software producers.

Deterrence. Finally, deterrence is considerably less effective in cyberspace than in the physical world. De-

terrence depends on being able to attribute acts to individuals or institutions and then punish the offenders.

- Attribution of attacks delivered over a network is difficult, because packets are relayed through multiple intermediaries and, therefore, purported sources can be spoofed or rewritten along the way. Attribution thus requires time-consuming analysis of information beyond what might be available from network traffic.

- Punishment can be problematic because attackers can work outside the jurisdiction of the government where their target is located. To limit or monitor all traffic that is destined to the hosts within some government's jurisdiction can interfere with societal values such as openness and access to information. Such monitoring also is infeasible, given today's network architecture.

Making Progress

The time is ripe to be having discussions about investment and government interventions in support of cybersecurity. How much should we invest? And how should we resolve trade-offs that arise between security and (other) societal values? It will have to be national dialogue. Whether or not computer scientists lead, they need to be involved. And just as there is unlikely to be a single magic-bullet technology for making systems secure, there is unlikely to be a magic-bullet intervention to foster the needed investments. □

Reference

1. Vardi, M. Cyber insecurity and cyber libertarianism. *Commun. ACM* 60, 5 (May 2017), 5.

Fred B. Schneider (fbs@cs.cornell.edu) Fred B. Schneider is Samuel B. Eckert Professor of Computer Science and chair of the at Cornell University computer science department, Cornell University, USA.

The impetus for this Viewpoint was a series of discussions with *Communications* Senior Editor Moshe Vardi during the two years preceding his May 2017 *Communications* Editor's Letter. Susan Landau, Lyn Millett, and Deirdre Mulligan read an earlier version of this Viewpoint and provided helpful and timely feedback. I am also grateful to the two reviewers for their comments, which resulted in this Viewpoint having a better-defined focus.

The author's work has been supported in part by AFOSR grant F9550-16-0250 and NSF grant 1642120. The views and conclusions contained in this Viewpoint are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of these organizations or the U.S. government.

Copyright held by author.

DOI:10.1145/3180492

M.S. Silberman, B. Tomlinson, R. LaPlante, J. Ross, L. Irani, and A. Zaldivar

Viewpoint

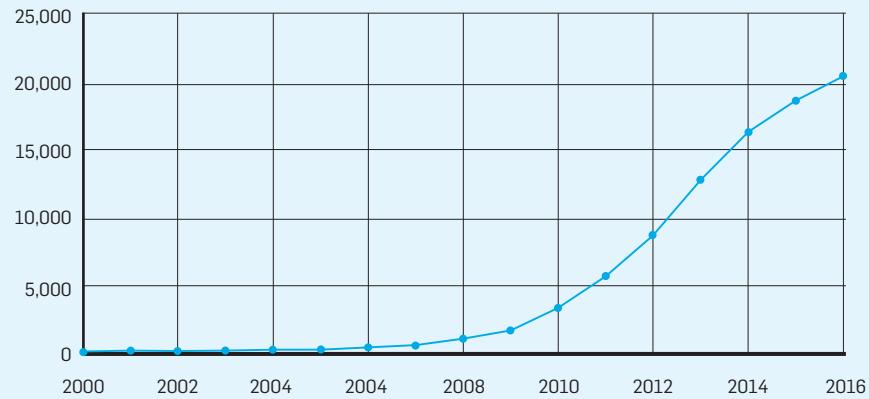
Responsible Research with Crowds: Pay Crowdworkers at Least Minimum Wage

High-level guidelines for the treatment of crowdworkers.

CROWDSOURCING IS INCREASINGLY important in scientific research. According to Google Scholar, the number of papers including the term “crowdsourcing” has grown from less than 1,000 papers per year pre-2008 to over 20,000 papers in 2016 (see the accompanying figure).

Crowdsourcing, including crowd-sourced research, is not always conducted responsibly. Typically this results not from malice but from misunderstanding or desire to use funding efficiently. Crowdsourcing platforms are complex; clients may not fully understand how they work. Workers’ relationships to crowdwork are diverse—as are their expectations about appropriate client behavior. Clients may be unaware of these expectations. Some platforms prime clients to expect cheap, “frictionless” completion of work without oversight, as if the platform were not an interface to human workers but a vast computer without living expenses. But researchers have learned that workers are happier and produce better work when clients pay well, respond to worker inquiries, and communicate with workers to improve task designs and quality control processes.⁶ Workers have varied but undervalued or unrecognized expertise and skills. Workers on Amazon’s Mechanical Turk platform (“MTurk”), for example, are more educated than the average U.S. worker.²

Papers in Google Scholar that use the term “crowdsourcing.”



Many advise clients on task design through worker forums. Workers’ skills offer researchers an opportunity to shift perspective, treating workers not as interchangeable subjects but as sources of insight that can lead to better research. When clients do not understand that crowdsourcing work, including research, involves interacting through a complex, error-prone system with human workers with diverse needs, expectations, and skills, they may unintentionally underpay or mistreat workers.

On MTurk, for example, clients may refuse to pay for (“reject”) completed work for any reason. Rejection exists to prevent workers from cheating—for example, completing a survey with random answers. But rejection also has a secondary usage: the percentage of

tasks a worker has had “approved”—that is, the percentage of tasks their clients chose to pay for—is interpreted as a proxy for worker quality, and used to automatically screen workers for tasks. A worker’s “approval rate,” however, can be negatively affected by client errors in quality control, compromising workers’ eligibility for other tasks. MTurk offers workers no way to contest rejections and no information about a client’s rejection history. Clients can screen workers based on a form of “reputation,” but not the reverse.

These dynamics seem especially relevant for workers who rely on crowdwork as a primary or significant secondary source of income. While some readers may be surprised to hear that people earn a living through

ACM Transactions on Reconfigurable Technology and Systems

ACM TREATS is a peer-reviewed and archival journal that covers reconfigurable technology, systems, and applications on reconfigurable computers. Topics include all levels of reconfigurable system abstractions and all aspects of reconfigurable technology including platforms, programming environments and application successes.



For further information or to submit your manuscript, visit trets.acm.org

crowdwork, research shows this is increasingly common, even in rich countries. In a 2015 International Labour Organization survey of MTurk workers (573 U.S. respondents), 38% of U.S. respondents said crowdwork was their primary source of income, with 40% of these (15% of U.S. respondents) reporting crowdwork as their *only* source of income.² In a 2016 Pew survey of 3,370 MTurk workers, 25% of U.S. respondents said that MTurk specifically was the source of “all or most” of their income.⁵

While it is to our knowledge generally not possible to be certain how representative any survey of crowdworkers is, these findings are consistent with both other MTurk-specific research and recent national surveys of online labor platform activity broadly—which includes “microtasking” platforms (such as MTurk), platforms for in-person work (such as Uber), and platforms for remote work (such as Upwork). For example, Farrell and Greig³ found that overall the “platform economy was a secondary source of income,” but that “as of September 2015, labor platform income represented more than 75% of total income for 25% of active [labor platform] participants,” or approximately 250,000 workers.^a

With crowdwork playing an economically important role in the lives of hundreds of thousands—or millions—of people worldwide, we ask: What are the responsibilities of clients and platform operators?

Crowdsourcing is currently largely “outside the purview of labor laws”⁸—but only because most platforms classify workers as “independent contractors,” not employees. “Employees” in the U.S. are entitled to the protections of the Fair Labor Standards Act—minimum wage

^a Farrell and Greig³ report that 0.4% of adults “actively participate in” (receive income from) labor platforms each month. (“Labor platforms” here include both platforms for in-person work such as Uber as well as platforms for remote work such as MTurk and Upwork.) Per the CIA World Factbook, the U.S. total population is 321,369,000, with approximately 80.1% “adult” (“15 years or older”). Therefore the number of U.S. adults earning more than 75% of their income from labor platforms is approximately $0.25 * 0.004 * 0.801 * 321369000$, or 257,415. “Adults” is interpreted by Farrell and Greig as “18 years or older,” not “15 years or older,” so we round down to 250,000.

and overtime pay—but contractors are not. (Many countries have similar distinctions.) While this legal classification is unclear and contested, and there is growing recognition that at least some crowdworkers should receive many or all protections afforded employees (including Salehi et al.,⁹ Michelucci and Dickinson,⁸ and Berg²), these intentions have not yet been realized.

Our own research, which we have asked researchers to stop citing¹⁰ and will therefore not cite here, has been used to justify underpayment of workers. Reporting on MTurk demographics in 2008–2009, we reported that workers responding to our survey earned on average less than \$2/hour. This figure has been cited by researchers to justify payment of similar wages.

Our (now outdated) descriptive research, which reported averages from a sizable but not necessarily representative sample of MTurk workers, was *not* an endorsement of that wage. Additionally, *eight years* have passed since that study—it should not be used to orient current practice.

Therefore, we build on a long-running conversation in computing research on ethical treatment of crowdworkers (for example, Bederson and Quinn¹) by offering the following high-level guidelines for the treatment of paid crowdworkers in research.

Pay workers at least minimum wage at your location. Money is the primary motivation for most crowdworkers (see, for example, Litman et al.⁶ for MTurk). Most crowdworkers thus relate to paid crowdwork primarily as work, rather than as entertainment or a hobby; indeed, as noted previously, a significant minority rely on crowdwork as a primary income source. Most developed economies have set minimum wages for paid work; however, the common requirement (noted earlier) that workers agree to be classified as independent contractors allows workers to be denied the protections afforded employees, including minimum wage.

Ethical conduct with respect to research subjects often requires researchers to protect subjects beyond the bare minimum required by law; given the importance of money as a motivation for most crowdworkers, it is ethically appropriate to pay crowdworkers minimum wage. Further, workers have

To make crowdsourced research possible, researchers and IRBs must develop ongoing, respectful dialogue with crowdworkers.

requested this (Salehi et al.⁹). Ethics demands we take worker requests seriously.

While crowdworkers are often located around the world, minimum wage at the client's location is a defensible lower limit on payment. If workers are underpaid, for example, due to underestimation of how long a task might take, correct the problem (for instance, on MTurk, with bonuses). On MTurk, if workers are refused payment mistakenly, reverse the rejections to prevent damage to the workers' approval rating. Note that fair wages lead to higher quality crowdsourced research.⁶

Remember you are interacting with human beings, some of whom complete these tasks for a living. Treat them at least as well as you would treat an in-person co-worker. As workers themselves have gone to great lengths to express to the public,⁴ crowdworkers are not interchangeable parts of a vast computing system, but rather human beings who must pay rent, buy food, and put children through school—and who have, just like clients, career and life goals and the desire to be acknowledged, valued, and treated with respect.

Respond quickly, clearly, concisely, and respectfully to worker questions and feedback via both email and worker forums (for example, turkernation.com, mturkcrowd.com). In addition to being a reasonable way to engage with human workers, this engagement may also improve the quality of the work you receive, since you may be informed of task design problems before a great deal of work has been done—and before you have incurred a responsibility to pay for that work, which was done in good faith.

Learn from workers. If workers tell you about technical problems or unclear instructions, address them promptly, developing workarounds as needed for workers who have completed the problematic task. Especially if you are new to crowdsourcing, you may unknowingly be committing errors or behaving inappropriately due to your study design or mode of engagement. Many workers have been active for years, and provide excellent advice. Workers communicate with one another and with clients in forums (as described earlier); MTurk workers in particular have articulated best practices for ethical research in the Dynamo Guidelines for Academic Requesters (guidelines.wearedynamo.org; Salehi et al.⁹).

Currently, the design of major crowdsourcing platforms makes it difficult to follow these guidelines. Consider a researcher who posts a task to MTurk, and after the task is posted, discovers that even expert workers take twice as long as expected. This is unsurprising; recent research shows that task instructions are often unclear to workers. If this researcher wishes to pay workers "after-the-fact" bonuses to ensure they are paid the intended wage, this can only be done one-by-one or with command-line tools. The former is time-consuming and tedious; the latter is only usable for a relative minority of clients. The platform's affordances (or non-affordances) are powerful determiners of how clients (are able to) treat workers. We suggest platform operators would do workers, clients, and themselves a service by making it easier for clients to treat workers well in these cases.

Finally, we call on university Institutional Review Boards to turn their attention to the question of responsible crowdsourced research. Crowdworkers relate to their participation in crowdsourced research primarily as *workers*. Thus the relation between researchers and crowdworkers is markedly different than researchers' relation to study participants from other "pools." While there may be some exceptions, we thus believe researchers should generally pay crowdworkers at least minimum wage. We urge IRBs to consider this position.

These suggestions are a start, not a comprehensive checklist. To make

crowdsourced research responsible, researchers and IRBs must develop ongoing, respectful dialogue with crowdworkers.

Further Reading

For detailed treatment of ethical issues in crowdwork, see Martin et al.⁷ For alternatives to MTurk, see Vakharia and Lease¹¹ or type "mturk alternatives" into any search engine. Readers interested in ethical design of labor platforms should seek recent discussions on "platform cooperativism" (for example, platformcoop.net). □

References

1. Bederson, B. and Quinn, A.J. Web workers unite! Addressing challenges of online laborers. In *Proceedings of CHI '11 EA* (2011), 97–106.
2. Berg, J. Income security in the on-demand economy: Findings and policy lessons from a survey of crowdworkers. *Comparative Labor Law & Policy Journal* 37, 3 (2016).
3. Farrell, D. and Greig, F. Paychecks, paydays, and the online platform economy: Big data on income volatility. JP Morgan Chase Institute, 2016.
4. Harris, M. Amazon's Mechanical Turk workers protest: 'I am a human being, not an algorithm.' *The Guardian* (Dec. 3, 2014); <http://bit.ly/2EcZVMS>.
5. Hitlin, P. Research in the crowdsourcing age, a case study. Pew Research Center, July 2016.
6. Litman, L., Robinson, J., and Rosenzweig, C. The relationship between motivation, monetary compensation, and data quality among U.S.- and India-based workers on Mechanical Turk. *Behavior Research Methods* 47, 2 (Feb. 2015), 519–528.
7. Martin, D. et al. Turking in a global labour market. *Computer Supported Cooperative Work* 25, 1 (Jan. 2016), 39–77.
8. Michelucci, P. and Dickinson, J.L. The power of crowds. *Science* 351, 6268 (2016), 32–33.
9. Salehi, N. et al., Eds. Guidelines for Academic Requesters—WeAreDynamo Wiki (2014); <http://bit.ly/1q6pY33>.
10. Silberman, M. et al. Stop citing Ross et al. 2010, 'Who are the crowdworkers?'; <http://bit.ly/2FkrObs>.
11. Vakharia, D. and Lease, M. Beyond Mechanical Turk: An analysis of paid crowd work platforms. In *Proceedings of iConference 2015*. (2015).

M. Six Silberman (michael.silberman@igmetall.de) works in the Crowdsourcing Project at IG Metall, 60329 Frankfurt am Main, Germany.

Bill Tomlinson (wmt@ics.uci.edu) is a Professor in the Department of Informatics at the University of California, Irvine, CA, USA, and a Professor in the School of Information Management, Victoria University of Wellington, New Zealand.

Rochelle LaPlante (rochelle@rochellelaplante.com) is a professional crowdworker, Seattle, WA, USA.

Joel Ross (joelross@uw.edu) is a Senior Lecturer in the Information School, University of Washington, Seattle, WA, USA.

Lilly Irani (lirani@ucsd.edu) is an Assistant Professor in the Communication Department and Science Studies Program, University of California, San Diego, CA, USA.

Andrew Zaldivar (azaldiva@uci.edu) is a Researcher in the Department of Cognitive Sciences, University of California, Irvine, CA, USA (now at Google).

This material is based upon work supported in part by National Science Foundation Grant CCF-1442749. The authors thank Janine Berg and Valerio De Stefano for comments. This Viewpoint reflects the authors' views, not any official organizational position.

Copyright held by authors.

Viewpoint

Computational Social Science ≠ Computer Science + Social Data

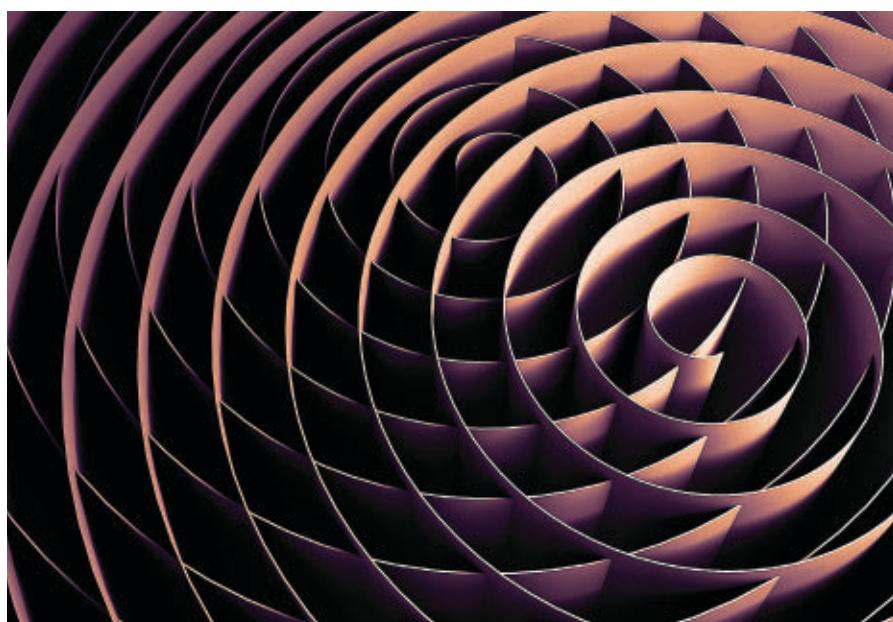
The important intersection of computer science and social science.

THIS VIEWPOINT IS about differences between computer science and social science, and their implications for computational social science. Spoiler alert: The punchline is simple. Despite all the hype, machine learning is not a be-all and end-all solution. We still need social scientists if we are going to use machine learning to study social phenomena in a responsible and ethical manner.

I am a machine learning researcher by training. That said, my recent work has been pretty far from traditional machine learning. Instead, my focus has been on computational social science—the study of social phenomena using digitized information and computational and statistical methods.

For example, imagine you want to know how much activity on websites such as Amazon or Netflix is caused by recommendations versus other factors. To answer this question, you might develop a statistical model for estimating causal effects from observational data such as the numbers of recommendation-based visits and numbers of total visits to individual product or movie pages over time.⁹

Alternatively, imagine you are interested in explaining when and why senators' voting patterns on particular issues deviate from what would be expected from their party affiliations and ideologies. To answer this question, you might model a set of issue-



based adjustments to each senator's ideological position using their congressional voting history and the corresponding bill text.^{4,8}

Finally, imagine you want to study the faculty hiring system in the U.S. to determine whether there is evidence of a hierarchy reflective of systematic social inequality. Here, you might model the dynamics of hiring relationships between universities over time using the placements of thousands of tenure-track faculty.³

Unsurprisingly, tackling these kinds of questions requires an interdisciplinary approach—and, indeed, computa-

tional social science sits at the intersection of computer science, statistics, and social science.

For me, shifting away from traditional machine learning and into this interdisciplinary space has meant that I have needed to think outside the algorithmic black boxes often associated with machine learning, focusing instead on the opportunities and challenges involved in developing and using machine learning methods to analyze real-world data about society.

This Viewpoint constitutes a reflection on these opportunities and challenges. I structure my discussion here

around three points—goals, models, and data—before explaining how machine learning for social science therefore differs from machine learning for other applications.

Goals

When I first started working in computational social science, I kept overhearing conversations between computer scientists and social scientists that involved sentences like, “I don’t get it—how is that even research?” And I could not understand why. But then I found this quote by Gary King and Dan Hopkins—two political scientists—that, I think, really captures the heart of this disconnect: “[C]omputer scientists may be interested in finding the needle in the haystack—such as [...] the right Web page to display from a search—but social scientists are more commonly interested in characterizing the haystack.”⁶

In other words, the conversations I kept overhearing were occurring because the goals typically pursued by computer scientists and social scientists fall into two very different categories.

The first category is prediction. Prediction is all about using observed data to reason about missing information or future, yet-to-be-observed data. To use King and Hopkins’ terminology, these are “finding the needle” tasks. In general, it is computer scientists and decision makers who are most interested in them. Sure enough, machine learning has traditionally focused on prediction tasks—such as classifying images, recognizing handwriting, and playing games like chess and Go.

The second category is explanation. Here the focus is on “why” or “how” questions—in other words, finding plausible explanations for observed data. These explanations can then be compared with established theories or previous findings, or used to generate new theories. Explanation tasks are therefore “characterizing the haystack” tasks and, in general, it is social scientists who are most interested in them. As a result, social scientists are trained to construct careful research questions with clear, testable hypotheses. For example, are women consistently excluded from long-term strategic planning in the workplace? Are government organizations more likely to comply with a public records request if they know that their peer organizations have already complied?

The goals typically pursued by computer scientists and social scientists fall into two very different categories.

Models

These different goals—prediction and explanation—lead to very different modeling approaches. In many prediction tasks, causality plays no role. The emphasis is firmly on predictive accuracy. In other words, we do not care why a model makes good predictions; we just care that it does. As a result, models for prediction seldom need to be interpretable. This means that there are few constraints on their structure. They can be arbitrarily complex black boxes that require large amounts of data to train. For example, GoogLeNet, a “deep” neural network, uses 22 layers with millions of parameters to classify images into 1,000 distinct categories.¹⁰

In contrast, explanation tasks are fundamentally concerned with causality. Here, the goal is to use observed data to provide evidence in support or opposition of causal explanations. As a result, models for explanation must be interpretable. Their structure must be easily linked back to the explanation of interest and grounded in existing theoretical knowledge about the world. Many social scientists therefore use models that draw on ideas from Bayesian statistics—a natural way to express prior beliefs, represent uncertainty, and make modeling assumptions explicit.⁷

To put it differently, models for prediction are often intended to *replace* human interpretation or reasoning, whereas models for explanation are intended to *inform* or *guide* human reasoning.

Data

As well as pursuing different goals, computer scientists and social scientists typically work with different types of data. Computer scientists usually work with large-scale, digitized data-

Calendar of Events

March 5–8

HRI ‘18: ACM/IEEE International Conference on Human-Robot Interaction
Chicago, IL,
Contact: Takayuki Kanda,
Email: kanda@atr.jp

March 7–11

IUI’18: 23rd International Conference on Intelligent User Interfaces
Tokyo, Japan,
Co-Sponsored: ACM/SIG,
Contact: Shlomo Berkovsky,
Email: shlomo.berkovsky@csiro.au

March 11–15

CHIIR ‘18: Conference on Human Information Interaction and Retrieval
New Brunswick, NJ,
Sponsored: ACM/SIG,
Contact: Chirag Shah,
Email: chirags@rutgers.edu

March 15

AID ‘18: AI Decentralized 2018
Toronto, ON, Canada,
Sponsored: ACM/SIG,
Contact: Toufi Saliba,
Email: toufi@privacyshell.com

March 15–16

TAU ‘18: ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems
Monterey, CA,
Sponsored: ACM/SIG,
Contact: Athanasius Spyrou,
Email: tom@e-spyrou.com

March 18–21

TEI ‘18: 12th International Conference on Tangible, Embedded, and Embodied Interaction
Stockholm, Sweden,
Sponsored: ACM/SIG,
Contact: Martin Jonsson,
Email: martin.jonsson@sh.se

March 19–21

CODASPY ‘18: 7th ACM Conference on Data and Application Security and Privacy
Tempe, AZ,
Sponsored: ACM/SIG,
Contact: Gail-Joon Ahn,
Email: gahn@asu.edu

March 19–23

DATE ‘18: Design, Automation and Test in Europe
Dresden, Germany,
Contact: Jan Madsen,
Email: jama@dtu.dk

sets, often collected and made available for no particular purpose other than “machine learning research.” In contrast, social scientists often use data collected or curated in order to answer specific questions. Because this process is extremely labor intensive, these datasets have traditionally been small scale.

But—and this is one of the driving forces behind computational social science—thanks to the Internet, we now have all kinds of opportunities to obtain large-scale, digitized datasets that document a variety of social phenomena, many of which we had no way of studying previously. For example, my collaborator Bruce Desmarais and I wanted to conduct a data-driven study of local government communication networks, focusing on how political actors at the local level communicate with one another and with the general public. It turns out that most U.S. states have sunshine laws that mimic the federal Freedom of Information Act. These laws require local governments to archive textual records—including, in many states, email—and disclose them to the public upon request.

Desmarais and I therefore issued public records requests to the 100 county governments in North Carolina, requesting all non-private email messages sent and received by each county’s department managers during a randomly selected three-month time frame. Out of curiosity, we also decided to use the process of requesting these email messages as an opportunity to conduct a randomized field experiment to test whether county governments are more likely to fulfill a public records request when they are aware that their peer governments have already fulfilled the same request.

On average, we found that counties who were informed that their peers had already complied took fewer days to acknowledge our request and were more likely to actually fulfill it. And we ended up with over half a million email messages from 25 different county governments.²

Challenges

Clearly, new opportunities like this are great. But these kinds of opportunities also raise new challenges. Most conspicuously, it is very tempting to

say, “Why not use these large-scale, social datasets in combination with the powerful predictive models developed by computer scientists?” However, unlike the datasets traditionally used by computer scientists, these new datasets are often about people going about their everyday lives—their attributes, their actions, and their interactions. Not only do these datasets document social phenomena on a massive scale, they often do so at the granularity of individual people and their second-to-second behavior. As a result, they raise some complicated ethical questions regarding privacy, fairness, and accountability.

It is clear from the media that one of the things that terrifies people the most about machine learning is the use of black-box predictive models in social contexts, where it is possible to do more harm than good. There is a great deal of concern—and rightly so—that these models will reinforce existing structural biases and marginalize historically disadvantaged populations.

In addition, when datapoints are humans, error analysis takes on a whole new level of importance because errors have real-world consequences that involve people’s lives. It is not enough for a model to be 95% accurate—we need to know who is affected when there is a mistake, and in what way. For example, there is a substantial difference between a model that is 95% accurate because of noise and one that is 95% accurate because it performs perfectly for white men, but achieves only 50% accuracy when making predictions about women and minorities. Even with large datasets, there is always proportionally less data available about minorities, and statistical patterns that hold for the majority may be invalid for a given minority group. As a result, the usual machine learning objective of “good performance on average,” may be detrimental to those in a minority group.^{1,5}

Thus, when we use machine learning to reason about social phenomena—and especially when we do so to draw actionable conclusions—we have to be exceptionally careful. More so than when we use machine learning in other contexts. But here is the thing: these ethical challenges are not entirely new. Sure, they may be

new to most computer scientists, but they are not new to social scientists.

Conclusion

To me, then, this highlights an important path forward. Clearly, machine learning is incredibly useful—and, in particular, machine learning is useful for social science. But we must treat machine learning for social science very differently from the way we treat machine learning for, say, handwriting recognition or playing chess. We cannot just apply machine learning methods in a black-box fashion, as if computational social science were simply computer science plus social data. We need transparency. We need to prioritize interpretability—even in predictive contexts. We need to conduct rigorous, detailed error analyses. We need to represent uncertainty. But, most importantly, we need to work with social scientists in order to understand the ethical implications and consequences of our modeling decisions. C

References

1. Barocas, S. and Selbst, A.D. Big data’s disparate impact. *California Law Review* 104 (2016), 671–732.
2. ben-Aaron, J. et al. Transparency by conformity: A field experiment evaluating openness in local governments. *Public Administration Review* 77, 1 (Jan. 2017), 68–77.
3. Clauset, A., Arbesman, S., and Larremore, D.B. Systematic inequality and hierarchy in faculty hiring networks. *Science Advances* 1, 1 (Jan. 2015).
4. Gerrish, S. and Blei, D. How they vote: Issue-adjusted models of legislative behavior. In *Advances in Neural Information Processing Systems Twenty Five* (2012), 2753–2761.
5. Hardt, H. How big data is unfair; <http://bit.ly/1BBgILr>.
6. Hopkins, D.J. and King, G. A method of automated nonparametric content analysis for social science. *American Journal of Political Science* 54, 1 (Jan. 2010), 229–247.
7. Jackman, S. *Bayesian Analysis for the Social Sciences*. Wiley, 2009.
8. Lauderdale, B.E. and Clark, T.S. Scaling politically meaningful dimensions using texts and votes. *American Journal of Political Science* 58, 3 (Mar. 2014), 754–771.
9. Sharma, A., Hofman, J., and Watts, D. Estimating the causal impact of recommendation systems from observational data. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation* (2015), 453–470.
10. Szegedy, C. et al. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015).

Hanna Wallach ([hanna@dirichlet.net](http://dirichlet.net)) is a Senior Researcher at Microsoft Research and an Adjunct Associate Professor at the University of Massachusetts Amherst.

This article is based on an essay that appeared on Medium—see <http://bit.ly/13QlExf>. This work was supported in part by NSF grant #IIS-1320219. Any opinions, findings and conclusions, or recommendations expressed in this material are those of the author and do not necessarily reflect those of the sponsor.

Copyright held by author.

ACM Welcomes the Colleges and Universities Participating in ACM's Academic Department Membership Program

ACM now offers an Academic Department Membership option, which allows universities and colleges to provide ACM Professional Membership to their faculty at a greatly reduced collective cost.

The following institutions currently participate in ACM's Academic Department Membership program:

- Appalachian State University
- Armstrong State University
- Ball State University
- Berea College
- Bryant University
- Calvin College
- Colgate University
- Colorado School of Mines
- Edgewood College
- Franklin University
- Georgia Institute of Technology
- Governors State University
- Harding University
- Hofstra University
- Howard Payne University
- Indiana University Bloomington
- Mount Holyoke College
- Northeastern University
- Ohio State University
- Old Dominion University
- Pacific Lutheran University
- Pennsylvania State University
- Regis University
- Roosevelt University
- Rutgers University
- Saint Louis University
- San José State University
- Shippensburg University
- St. John's University
- Trine University
- Trinity University
- Union College
- Union University
- University of California, Riverside
- University of Colorado Denver
- University of Connecticut
- University of Illinois at Chicago
- University of Jamestown
- University of Memphis
- University of Nebraska at Kearney
- University of Nebraska Omaha
- University of North Dakota
- University of Puget Sound
- University of the Fraser Valley
- University of Wyoming
- Virginia Commonwealth University
- Wake Forest University
- Wayne State University
- Western New England University
- Worcester State University

Through this program, each faculty member receives all the benefits of individual professional membership, including *Communications of the ACM*, member rates to attend ACM Special Interest Group conferences, member subscription rates to ACM journals, and much more.

For more information: www.acm.org/academic-dept-membership



Association for
Computing Machinery

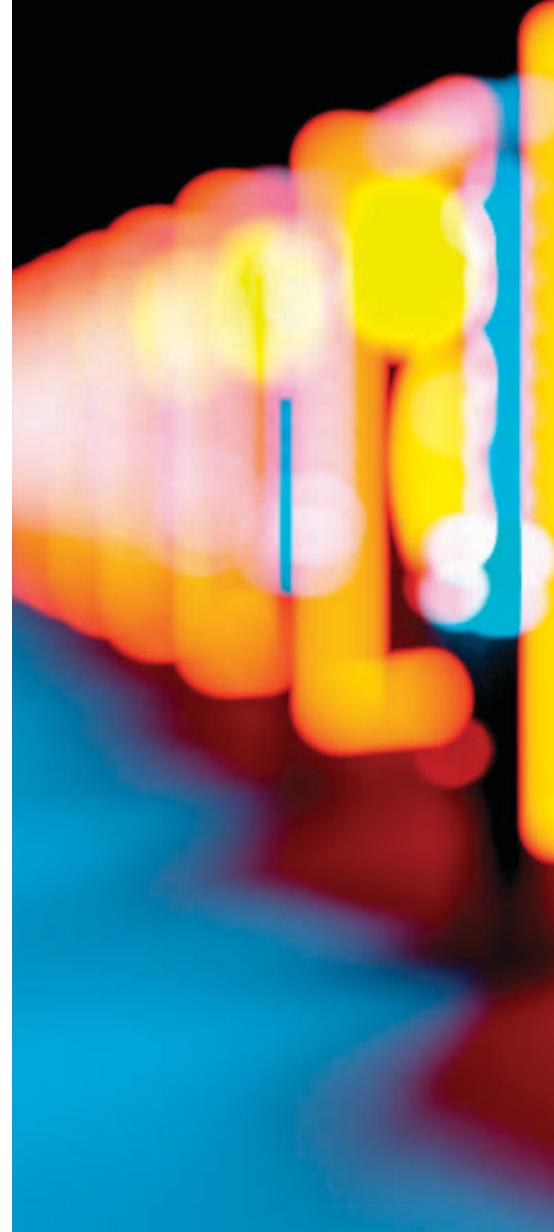
The unseen economic forces that govern the Bitcoin protocol.

BY YONATAN SOMPOLINSKY AND AVIV ZOHAR

Bitcoin's Underlying Incentives

BEYOND ITS ROLE as a protocol for managing and transferring money, the Bitcoin protocol creates a complex system of economic incentives that govern its inner workings. These incentives strongly impact the protocol's capabilities and security guarantees, and the path of its future development. This article explores these economic undercurrents, their strengths and flaws, and how they influence the protocol.

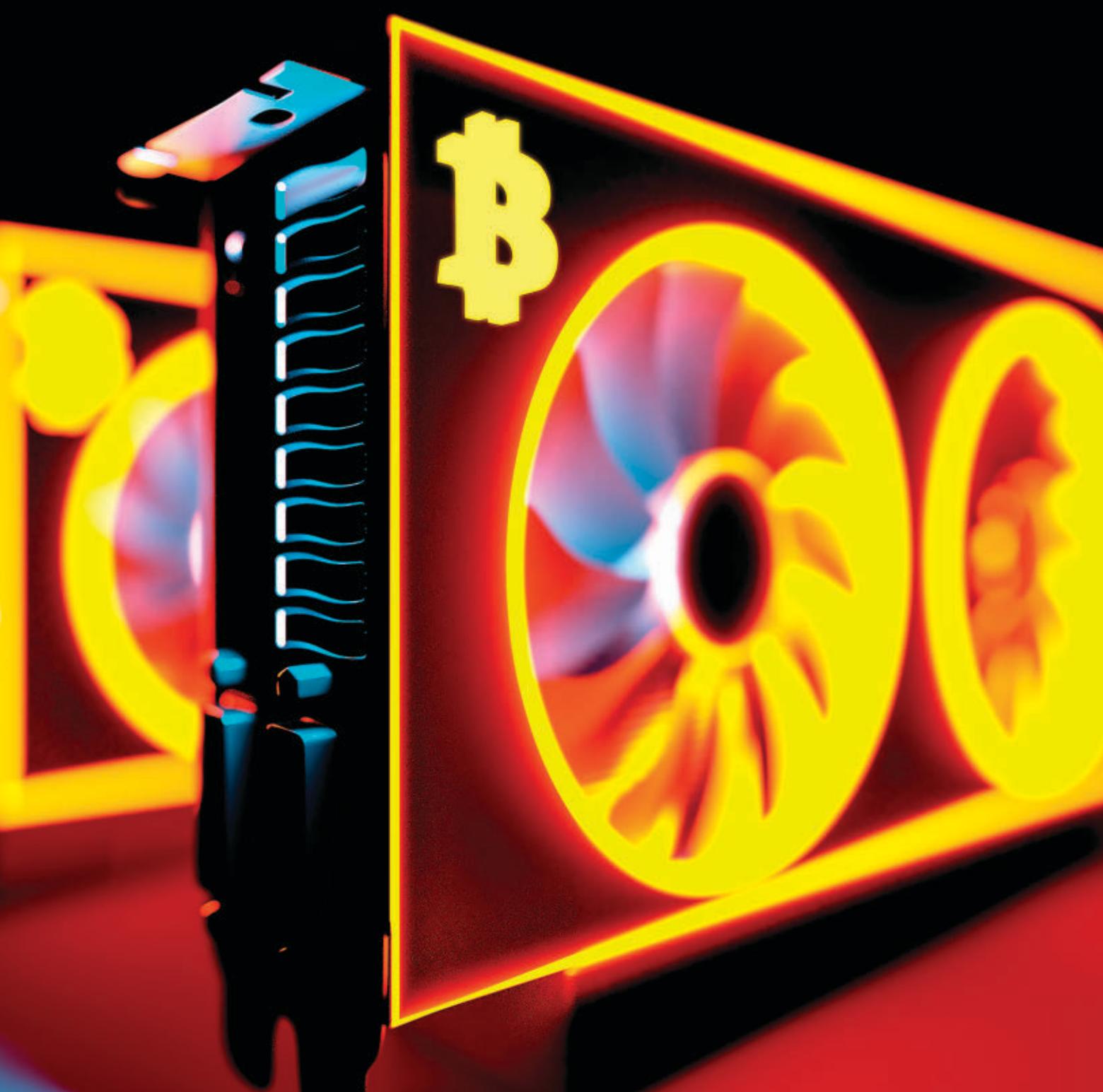
Bitcoin, which continues to enjoy growing popularity, is built upon an open peer-to-peer (P2P) network of nodes.⁹ The Bitcoin system is “permissionless”—anyone can choose to join the network, transfer money, and even participate in the authorization of transactions. Key to Bitcoin's security is its resilience to manipulations by attackers who may choose to join the system under multiple false identities. After all, anyone can download the open-source code for a Bitcoin node and add as many computers to this network as they like, without having to identify themselves to others. To counter



this, the protocol requires nodes that participate in the system to show proof that they exerted computational effort to solve hard cryptographic puzzles (proof-of-work) in order to participate actively in the protocol.

Nodes that engage in such work are called *miners*. The system rewards miners with bitcoins for generating proof-of-work, and thus sets the incentives for such investment of efforts.

The first and most obvious effect of participants getting paid in bitcoins for



running software on their computers was that, once bitcoins had sufficient value, people started mining quite a lot. In fact, efforts to mine intensified to such a degree that most mining quickly transitioned to dedicated computer farms that used specialized gear for this purpose: first, GPUs that were used to massively parallelize the work; and later, custom-designed chips, or ASICs (application-specific integrated circuits), tailored for the specific computation at the core of

the protocol (machines with current ASICs are about a million times faster than regular PCs when performing this work). The Bitcoin network quickly grew and became more secure, and competition for the payments given out periodically by the protocol became fierce.

Before discussing the interplay between Bitcoin's security and its economics, let's quickly look at the rules of the protocol itself; these give birth to this complex interplay.

A Quick Primer on the Bitcoin Protocol

Users who hold bitcoins and wish to transfer them send transaction messages (via software installed on their computer or smartphone) to one of the nodes on the Bitcoin network. Active nodes collect such transactions from users and spread them out to their peers in the network, each node informing other nodes it is connected to about the requested transfer. Transactions are then aggregated in batches called *blocks*.

Blocks, in turn, are chained together to create the *blockchain*, a record of all accepted bitcoin transactions. Each block in the chain references its predecessor block by including a cryptographic hash of that block—effectively a unique identifier of that predecessor. A complete copy of the blockchain is kept at every node in the Bitcoin network. The process of block creation is called *mining*. One of its outcomes (among others) is the printing of fresh coins, which we call *minting*.

The rules of the protocol make block creation extremely difficult; a block is considered legal only if it contains the answer to a hard cryptographic puzzle. As compensation, whenever miners manage to create blocks they are rewarded with bitcoins. Their reward is partly made up of newly minted bitcoins and partly of mining fees collected from all of the transactions embedded in their blocks. The rate of minting is currently 12.5 bitcoins per block. This amount is halved approximately every four years. As this amount decreases, Bitcoin begins to rely more and more on transaction fees to pay the miners.

The key to Bitcoin's operation is to get all nodes to agree on the contents of the blockchain, which serves as the record of all transfers in the system. Blocks are thus propagated quickly to all nodes in the network. Still, it is sometimes possible for nodes to receive two different versions of the blockchain. For example, if two nodes manage to create a block at the same time, they may hold two different extensions to the blockchain. These blocks might contain different sets of payments, and so a decision must be made on which version to accept.

The Bitcoin protocol dictates that nodes accept only the longest chain as the correct version of events, as shown in the accompanying figure. (To be more precise, nodes select the chain that contains the most accumulated computational work. This is usually the longest chain.) This rule, often called the “longest chain rule,” provides Bitcoin with its security. An attacker who wishes to dupe nodes into believing that a different set of payments has occurred will need to produce a longer chain than that of the rest of the network—a task that is incredibly dif-

**The key to
Bitcoin's operation
is to get all nodes
to agree on
the contents
of the blockchain,
which serves
as the record
of all transfers
in the system.**

ficult because of the proof-of-work required for each block's creation. In fact, as long as the attacker has less computational power than the entire Bitcoin network put together, blocks and transactions in the blockchain become increasingly harder to replace as the chain above them grows.

This difficulty in replacing the chain implies that it takes many attempts before an attacker can succeed in doing so. These failed attempts supposedly impose a cost on attackers—mining blocks off of the longest chain without getting the associated mining rewards. Naïve attacks are indeed costly for attackers (more sophisticated attacks are discussed later).

The accompanying figure on page 49 shows the evolution of the blockchain: forks appear and are resolved as one of the branches becomes longer than the other. Blocks that are off the longest chain are eventually abandoned. They are no longer extended, their contents (transactions colored in red) are ignored, and the miners that created them receive no reward. At point 1 there are two alternative chains resulting from the creation of a block that did not reference the latest tip of the blockchain. At point 2 the fork is resolved, as one chain is longer than the other. At point 3 there is another fork that lasted longer, and at point 4 the second fork is resolved.

Bitcoin Economics 101: Difficulty Adjustment and the Economic Equilibrium of Mining

Bitcoin's rate of block creation is kept roughly constant by the protocol: Blocks are created at random intervals of roughly 10 minutes in expectation. The difficulty of the proof-of-work required to generate blocks increases automatically if blocks are created too quickly. This mechanism has been put in place to ensure that blocks do not flood nodes as more computational power is added to the system. The system thus provides payments to miners at a relatively constant rate, regardless of the amount of computational power invested in mining.

Clearly, as the value (in U.S. dollars) of bitcoins rises, the mining business (which yields payments that are denominated in bitcoins) becomes more

lucrative. More participants then find it profitable to join the group of miners, and, as a consequence, the difficulty of block creation increases. With this increase in difficulty, mining blocks slowly becomes more expensive. In the ideal case, the system reaches equilibrium when the cost of block creation equals the amount of extracted rewards. In fact, mining will always be slightly profitable—mining is risky, and also requires an initial investment in equipment, and some surplus in the rewards must compensate for this. Hence, Bitcoin's security effectively adjusts itself to match its value: A higher value also implies higher security for the protocol.

As mining rewards continue to decline (as per the protocol's mining schedule), the incentive to create blocks is expected to rely more on transaction fees. If a sudden drop in bitcoin transaction volume occurs, these fees might be insufficient to compensate miners for their computational resources. Some miners might then halt their block creation process, temporarily. This may compromise the system, as the security of transactions depends on all honest miners actively participating. (For additional work on the incentives in Bitcoin after mining declines, see Carlsten et al.³)

Many complain that the computation required to create blocks wastes resources (especially electricity) and has no economic goal other than imposing large costs on would-be attackers of the system. The proof-of-work is indeed a solution to a useless cryptographic puzzle—except, of course, that this “useless” work secures the Bitcoin network. But what if some of the work could be useful? Or could be produced more efficiently? If mining does not entail a waste of resources for each node, then it also costs nothing for attackers to attack the system. In fact, if the proof-of-work is less costly to solve, more honest participants join mining (to collect the rewards), and soon the difficulty adjustment mechanism raises the difficulty again. Hence, in a sense, the Bitcoin proof-of-work is built to spend a certain amount of resources no matter how efficient an individual miner becomes. To derive substantial benefits from mining without an offsetting increase in costs re-

quires a proof-of-work that is useful to society at large but cannot provide value to the individual miner. (For some attempts at using other problems as a basis for proof-of-work, see Ball et al.,² Miller et al.,⁸ and Zhang et al.¹³)

Mining Decentralization

The key aspect of the Bitcoin protocol is its decentralization: no single entity has a priori more authority or control over the system than others. This promotes both the resilience of the system, which does not have a single anchor of trust or single point of failure, and competition among the different participants for mining fees.

To maintain this decentralization, it is important that mining activity in Bitcoin be done by many small entities and that no single miner significantly outweigh the others. Ideally, the rewards that are given to miners should reflect the amount of effort they put in: a miner who contributes an α -fraction of the computational resources should create an α -fraction of the blocks on average, and as a consequence extract a proportional α -fraction of all allocated fees and block rewards.

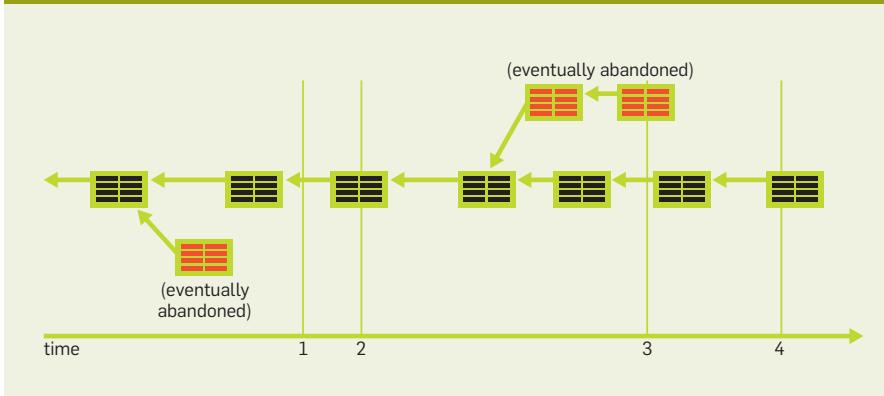
In practice, some participants can benefit disproportionately from mining, for several different reasons. An unbalanced reward allocation of this sort creates a bias in favor of larger miners with more computational power, making them more profitable than their smaller counterparts and creating a constant economic undercurrent toward the centralization of the system. Even slight advantages can endanger the system, as the miner can use additional returns to purchase more and more computational power, raising the difficulty of mining as the

miner grows and pushing the other smaller (and, hence, less profitable) miners out of the game. The resulting winner-takes-all dynamic inevitably leads to centralization within the system, which is then at the mercy of the prevailing miner, and no security properties can be guaranteed.

ASICs mining. The appearance of ASICs initially alarmed the Bitcoin community. ASICs were orders of magnitude more efficient at mining bitcoins than previous systems. As this special hardware was not initially easy to acquire, it provided its owners with a great advantage over other miners—they could mine at a much lower cost. Those with this advantage would add ASIC-based proof-of-work to the system until the difficulty level would be so high that everyone else would quit mining. The risk was then that a single large miner would have sole access to ASICs and would come to dominate the Bitcoin system. Concerns subsided after some time, as ASICs became commercially available and more widely distributed.

In fact, ASIC mining actually introduces long-term effects that contribute to security. Later this article looks at how a miner can carry out profitable double spending and selfish mining attacks. One can argue, however, that even selfish and strategic miners are better off avoiding such attacks. Indeed, a miner who invested millions of dollars in mining equipment such as ASICs is heavily invested in the future value of Bitcoin: the miner's equipment is expected to yield payments of bitcoins over a long period in the future. Should the miner then use this gear to attack the system, confidence in the currency would drop, and with

The evolution of the blockchain.



it the value of bitcoins and future rewards. The interests of miners are thus, in some sense, aligned with the overall health of the system.

All in all, ASIC mining introduces a *barrier-to-entry* to the system, as ordinary people cannot simply join the mining efforts; it thus reduces decentralization. On the other hand, it introduces a form of *barrier-to-exit*, as miners cannot repurpose their equipment to other economic activities; it therefore contributes to security.

The appearance of competing cryptocurrencies (for example, Litecoin, which essentially cloned Bitcoin), some of which use the same proof-of-work as Bitcoin, offers alternatives for miners who wish to divert their mining power elsewhere. This introduces complex market dynamics. For example, when a specific currency loses some value, miners will divert their mining power to another cryptocurrency until the difficulty readjusts. This can cause fluctuations in block creation that destabilize smaller cryptocurrencies.

Alternative systems with no ASIC mining. Interestingly, some cryptocurrencies use different proof-of-work puzzles that are thought to be more resistant to ASIC mining, that is, they choose puzzles for which it is difficult to design specialized hardware; for example, Ethereum uses the Ethash puzzle (<https://github.com/ethereum/wiki/wiki/Ethash>). This is often achieved by designing algorithmic problems that require heavy access to other resources, such as memory, and that can be solved efficiently by commercially available hardware.

These alternative systems are in principle more decentralized, but on the flip side they lack the barrier-to-exit effect and its contribution to security.

A similar effect occurs when cloud mining becomes highly available. Some mining entities offer their equipment for rental over the cloud. The clients of these businesses are effectively miners who do not have a long-term stake in the system. As such services become cheaper and more accessible, anyone can easily become a temporary miner, with similar effects on security.

ASICBoost. Recall that creating a block requires solving a cryptographic puzzle unique to that block. This involves guessing inputs to a crypto-

graphic hash function. Solving the puzzle is mostly done via brute-force enumeration of different inputs.

A miner can gain an advantage by creating blocks using more efficient methods than his or her counterparts. In addition to better hardware, an advantage can take a more algorithmic form. In fact, an algorithmic “trick” nicknamed ASICBoost has recently made headlines. ASICBoost enables the miner to reuse some of the computational work performed during the evaluation of one input for the evaluation of another. This algorithm is proprietary, patent pending, and it is unclear who is and who is not using it. Such an algorithmic advantage can be translated to lower power consumption per hash. Bitmain, a large manufacturer of ASICs for bitcoin mining that also operates some mining pools, was recently accused by some of secretly deploying a hardware variant of ASICBoost to increase its profits. Allegations were made that this company was politically blocking some protocol improvements that would coincidentally remove their ability to use ASICBoost.

Communication. Yet another method for a miner to become more efficient is to invest in communication infrastructure. By propagating blocks faster, and by receiving others' blocks faster, a miner can reduce the chances that their blocks will not belong to the longest chain and will be discarded (“orphaned”). As off-chain blocks receive no rewards, a better connection to the network translates to reduced losses. Admittedly, with Bitcoin's current block creation rate, this advantage is rather marginal; blocks are created infrequently, and speeding up delivery by just a few seconds yields relatively little advantage. Nonetheless, better connectivity is a relatively cheap way to become more profitable.

Furthermore, the effects of communication become much more pronounced when the protocol is scaled up and transaction processing is accelerated. Today, Bitcoin clears three to seven transactions per second on average. Changing the parameters of Bitcoin to process more transactions per second would increase the rate of orphan blocks and would amplify the advantage of well-connected miners.

Economies of scale. As with any

large entity, professional miners may enjoy the economic benefits of size. With a larger mining operation, such miners are much more likely to invest in different optimizations, such as finding sources of somewhat cheaper electricity, or placing their equipment in cooler regions to provide more efficient cooling to their machines (mining usually consumes a great deal of electricity, and cooling the machines presents a real challenge). Large miners can also purchase ASICs in bulk for better prices. All of this translates to natural advantages to size, a phenomenon that is not specific to Bitcoin but in fact appears in many industries. These effects give large miners an advantage and slowly pull the system toward a centralized one.

Many have raised concerns that most of today's Bitcoin mining is done by Chinese miners. They enjoy better access to ASICs, cheaper electricity, and somewhat lower regulation than similar operations in other locations. The Chinese government, which tightly controls Internet traffic in and out of China, could choose to disrupt the system or even seize the mining equipment that is within its borders.

Mining Pools and Risk Aversion

Bitcoin's mining process yields very high reward but with very low probability for each small miner. A single ASIC that is running full time may have less than a 1-in-600,000 chance of mining the next block, which implies that years can go by without finding a single block. This sort of high-risk/high-reward payoff is not suitable for most. Many would prefer a small, constant rate of income over long periods of time (this is essentially risk aversion).⁶ A constant income stream can be used, for example, to pay the electric bills for mining.

The formation of pools. Mining pools are coalitions of miners that combine their computational resources to create blocks together and share the rewards among members of the pool. Since the pool's workers together find blocks much more often than each miner alone, they are able to provide small continuous payments to each worker on a more regular basis.

From the perspective of the Bitcoin network, the pool is just a single min-

ing node. Pool participants interact with the pool's server, which sends the next block header that the pool is working on to all workers. Each member tries to solve the cryptographic puzzle corresponding to this block (in fact, they use small variants of the same block and work on slightly different proof-of-work puzzles to avoid duplicating work). Whenever a worker finds a solution, it is sent to the pool manager, who in turn publishes the block to the network. The block provides a reward to the pool, which the manager then distributes among all of the pool's workers (minus some small fee).

Reward distribution within pools and possible manipulations. Many pools are public and open to any willing participant. Obviously, such pools must take measures to ensure that only members who truly contribute to the pool's mining efforts enjoy a portion of the rewards. To that end, every pool member sends *partial solutions* of the proof-of-work to the pool—these are solutions that came “close” to being full blocks. Partial solutions are much more common than full solutions, and anyone working on the problem can present a steady stream of such attempts that fall short of the target. This indicates that the worker is indeed engaged in work, and can be used to assess the amount of computational power each worker dedicates to the pool. Pools thus reward workers in some proportion to the number of shares that they earn (a share is granted for every partial solution that is submitted).

Fortunately, a pool member who has found a valid solution to the puzzle cannot steal the rewards. The cryptographic puzzle depends on the block header, which is under the control of the pool's manager. It encodes a commitment to the contents of the block itself (via a cryptographic hash), including the recipient of the block's rewards. After finding a valid solution for a specific block header, one cannot tamper with the header without invalidating the solution.

Nonetheless, pools are susceptible to some manipulations by strategic miners:

Pool hopping. In the early days of Bitcoin, mining pools would simply divide the reward from the latest block

Mining pools are coalitions of miners that combine their computational resources to create blocks together and share the rewards among members of the pool.

among all workers in proportion to the number of partial solutions each worker submitted. The number of shares was measured from the previous block created by the same pool.

Some workers came up with a way to improve their rewards: if a pool was unlucky and did not find a block for a while, many partial solutions (shares) would accumulate. If a block was then found by the pool, its reward would be split among many shares. Working to generate additional shares is just as costly as before but yields low expected rewards for this very reason. Instead, the worker could just switch to another pool in which a block had been found more recently, and in which each additional share granted a higher expected reward. If many adopt this behavior, a pool that is temporarily unsuccessful should, in fact, be completely abandoned by all rational miners. Pool-hopping-resistant reward schemes were quickly developed and adopted by most mining pools.¹⁰

Block-withholding attacks. While a miner cannot steal the block reward of a successful solution, he or she can still deny the rewards from the rest of the miners in the pool. The miner can choose to submit only partial solutions to the pool's manager but discard all successful solutions. The miner thus receives a share of the rewards when others find a solution, without providing any actual contribution to the pool. Discarding the successful solution sabotages the pool, and causes a small loss of income to the attacker.

In spite of the losses to an attacker, in some situations it is worthwhile for mining pools to devote some of their own mining power to sabotage their competitors: the attacker pool infiltrates the victim pool by registering some of its miners as workers in the victim pool. These workers then execute a block-withholding attack. Careful calculations of the costs and rewards show that, in some scenarios (depending on the sizes of the attacker and victim pools), the attack is profitable.⁴ To prevent such schemes, a slight modification of the mining protocol has been proposed. In the modified version, workers would not be able to discern between partial and full solutions to the proof-of-work puzzle and would not be able to selectively withhold full solutions.

Eliminating pools. While pools are good for small miners, mitigating their risk and uncertainty, they introduce some centralization to the system. The pool operator is essentially controlling the combined computational resources of many miners and is therefore quite powerful. Some researchers proposed a technical modification to the mining protocol that undermines the existence of public pools altogether.⁷ Under this scheme, after finding a valid solution to the block, the pool member who mined it would still be able to redirect the rewards to themselves (without invalidating the solution). Assuming many miners would claim the rewards for themselves, pools would not be profitable and would therefore dissolve.

The Economics of Attacks and Deviations from the Rules

Earlier, this article described methods by which a miner can become more dominant within the protocol—both to profit more than his or her fair share and to generate more of the blocks in the chain. The methods discussed thus far do not violate any of the protocol's rules; in some sense, miners are *expected* to make the most of their hardware and infrastructure. This section discusses direct violations of the rules of the protocol that allow miners to profit at the expense of others. In a sense, the existence of such strategies implies that there is something fundamentally broken in the protocol's incentive structure: rational profit-maximizing participants will not follow it.

Informally, the protocol instructs any node to: validate every new message it receives (block/transaction); propagate all valid messages to its peers; broadcast its own new blocks immediately upon creation; and, build its new blocks on top of the longest chain known to that node. Attacks on the protocol correspond to deviations from one or more of these instructions.

Validation. A miner who does not validate incoming messages is vulnerable—the next block might include an invalid transaction that he or she did not verify, or reference an invalid predecessor block. Other nodes will then consider this new block as invalid and ignore it. This sets a clear incentive for miners to embed in their blocks only

The existence of selfish mining strategies implies that there is something fundamentally broken in the protocol's incentive structure: rational profit-maximizing participants will not follow it.

valid transactions and to validate every new block before accepting it.

Interestingly, despite this logic, sometimes miners mine on top of a block without fully validating it. This practice is known as *SPV mining* (simplified payment verification usually refers to the use of thin clients that do not read the full contents of blocks).

Why would miners engage in building on top of an unvalidated block? The answer again lies in incentives. Some miners apply methods to learn about the hash ID of a newly created block even before receiving its entire contents. One such method, known as spy-mining, involves joining another mining pool as a worker to detect block creation events. Even when the block is received, it takes time to validate the transactions it contains. During this time, the miner is aware that the blockchain is already longer by one block. Therefore, rather than letting the mining equipment lie idle until the block is validated, the miner decides to mine on top of it, under the assumption that it will most likely be valid. To avoid the risk that the next block will contain conflicts with the transactions of the unverified block, the miner does not embed new transactions in the next block, hoping still to collect the block reward.

There is indeed evidence that miners are taking this approach. First, some fraction of the blocks being mined is empty (even when many transactions are waiting to be approved). Another piece of evidence is related to an unfortunate incident that took place in July 2015. An invalid block was (unintentionally) mined due to a bug, and SPV miners added five additional blocks on top of it without validating. Of course, other validating miners rejected that block and any block that referenced it, resulting in a six-block-long fork in the network. Blocks that were discarded in the fork could have contained double-spent transactions.

This event shows the dangers of SPV mining: it lowers the security of Bitcoin and may trigger forks in the blockchain. Fortunately, miners have vastly improved the propagation and validation time of blocks, so SPV mining has less and less effect. The planned decline in the minted reward given to

empty blocks will also lower the incentive to engage in such behavior.

Transaction propagation. A second important aspect of the Bitcoin protocol pertains to information propagation: new transactions and blocks should be sent to all peers in the network. Here the incentive to comply with the protocol is not so clear. Miners may even have a disincentive to share unconfirmed transactions that have yet to be included in blocks, especially transactions that offer high fees.¹ Miners have strong incentive to keep such transactions to themselves until they manage to create a block. Sending a transaction to others allows them to snatch the reward it offers first. Thus far, most transaction fees have been relatively low, and there is no evidence that transactions with high fees are being withheld in this way.

Next, let's turn our attention to deviations from the mining protocol intended explicitly to manipulate the blockchain.

Selfish mining. Whenever a miner creates a new block, the protocol says it should be created on top of the longest chain the miner observes (that is, to reference the tip of the longest chain as its predecessor) and that the miner should send the new block immediately to network peers.

Unfortunately, a miner can benefit by deviating from these rules and acting strategically.^{5,11} The miner's general strategy is to withhold the blocks' publication and keep the extension of the public chain secret. Meanwhile, the public chain is extended by other (honest) nodes. The strategic miner publishes the chain only when the risk that it will not prevail as the longest chain is too high. When the miner does so, all nodes adopt the longer extension that the miner suddenly released, as dictated by the protocol, and they discard the previous public extension.

Importantly, this behavior increases the miner's *share* in the longest chain—meaning, it increases the *percentage* of blocks on the eventual longest chain that the miner generates. Recall that Bitcoin automatically adjusts the difficulty of the proof-of-work so as to keep the block creation rate constant. Thus, in the long run, a larger relative share of blocks in the chain translates to an increase in the miner's absolute rewards.

There is no definite method to verify whether miners are engaging in selfish mining or not. Given that very few blocks are orphaned, it seems like this practice has not been taken up, at least not by large miners (who would also have the most to gain from it). One way to explain this is that miners who attempt such manipulation over the long term may suffer loss to their reputation and provoke outrage by the community. Another explanation is that this scheme initially requires losing some of the selfish miner's own blocks, and it becomes profitable only in the long run (it takes around two weeks for the protocol to readjust the difficulty level).

Double spending is the basic attack against Bitcoin users: the attacker publishes a legitimate payment to the network, waits for it to be embedded in the blockchain and for the victim to confirm it, and then publishes a longer chain of blocks mined in secret that do not contain this payment. The payment is then no longer part of the longest chain and, effectively, "never happened."

This attack incurs a risk: the attacker could lose the rewards for his or her blocks if they do not end up in the longest chain. Surprisingly, and unfortunately, a persistent attacker can eliminate this risk by following more sophisticated attack schemes.¹² The idea is to abandon the attack frequently, publish the secret attack chain, and collect rewards for its blocks. By resetting the attack whenever the risk of losing block rewards is too high, the attacker can eliminate the attack cost and even be profitable in the long term. These schemes are in essence a combination of selfish mining and double-spending attacks.

Currently, double spending is not observed often in the network. This could be because executing a successful double spend is difficult, or because the very miners who could execute such attacks successfully also have a heavy stake in the system's reputation.

Conclusion

Incentives do indeed play a big role in the Bitcoin protocol. They are crucial for its security and effectively drive its daily operation. As argued here, miners go to extreme lengths to maximize their revenue and often find creative ways to do so that are sometimes at odds with the protocol.

Cryptocurrency protocols should be placed on stronger foundations of incentives. There are many areas left to improve, ranging from the very basics of mining rewards and how they interact with the consensus mechanism, through the rewards in mining pools, and all the way to the transaction fee market itself. C

Related articles on queue.acm.org

Research for Practice: Cryptocurrencies, Blockchains, and Smart Contracts

Arvind Narayanan and Andrew Miller
<http://queue.acm.org/detail.cfm?id=3043967>

Bitcoin's Academic Pedigree

Arvind Narayanan and Jeremy Clark
<http://queue.acm.org/detail.cfm?id=3136559>

Certificate Transparency

Ben Laurie
<http://queue.acm.org/detail.cfm?id=2668154>

References

- Babaioff, M. et al. On Bitcoin and red balloons. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, 2012, 56–73.
- Ball, M. et al. Proofs of Useful Work. IACR Cryptology ePrint Archive, 2017, 203.
- Carlsten, M. et al. On the instability of Bitcoin without the block reward. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*, 2016, 154–167.
- Eyal, I. The Miner's dilemma. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2015.
- Eyal, I., and Sirer, E.G. Majority is not Enough: Bitcoin mining is vulnerable. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, Berlin, 2014.
- Fisch, B.A., Pass, R., Shelat, A. Socially Optimal Mining Pools. arXiv preprint, 2017.
- Miller, A. et al. Nonoutsorceable scratch-off puzzles to discourage Bitcoin mining coalitions. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015.
- Miller, A. et al. Permacoin: Repurposing Bitcoin work for data preservation. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2014.
- Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system. Bitcoin.org, 2008; <https://bitcoin.org/bitcoin.pdf>.
- Rosenfeld, M. Analysis of Bitcoin Pooled Mining Reward Systems. arXiv preprint, 2011.
- Sapirshtein, A., Sompolinsky, Y. and Zohar, A. Optimal selfish mining strategies in Bitcoin. In *Proceedings of the International Conference on Financial Cryptography and Data Security*. Springer, Berlin, 2016.
- Sompolinsky, Y. and Zohar, A. Bitcoin's security model revisited. In *Proceedings of the International Joint Conference on Artificial Intelligence, Workshop on A.I. in Security*, 2017. Melbourne.
- Zhang, F. et al. REM: Resource-Efficient Mining for Blockchains. Cryptology ePrint Archive. <https://eprint.iacr.org/2017/179>.

Yonatan Sompolinsky is a Ph.D. student at the School of Computer Science and Engineering at the Hebrew University of Jerusalem. He is founding scientist of DAGLabs.

Aviv Zohar is a faculty member at the School of Computer Science and Engineering at the Hebrew University of Jerusalem, and a cofounder and chief scientist of QED-it. He has been researching the scalability, security, and underlying incentives of cryptocurrencies for several years.



Being funny is serious work.

BY THOMAS A. LIMONCELLI

Operational Excellence in April Fools' Pranks

AT 10:23 UTC on April 1, 2015, stackoverflow.com enabled an April Fools' prank called StackEgg.¹ It was a simple Tamagotchi-like game that appeared in the upper-right corner of the company's website. Though it had been tested, we did not account for the additional network activity it would generate. By 13:14 UTC the activity had grown to the point of overloading the company's load balancers, making the site unusable. All of the company's Web properties were affected. The prank had, essentially, created a self-inflicted denial-of-service attack.

The engineers involved in the prank didn't panic. They went to a control panel and disabled the feature. Network activity returned to normal, and the site was operating again by 13:47 UTC. The problem was diagnosed, fixed, and new code was pushed into production by 14:56 UTC. The prank was saved!

Was Stack Overflow lucky that the engineers had designed the prank so that it could be easily disabled? No, it was not luck. It was all in the playbook for operational excellence in AFPs (April Fools' pranks).

A successful AFP depends on many operational best practices. In this article, I will share some of the key ones.

What Makes an April Fools' Prank Funny?

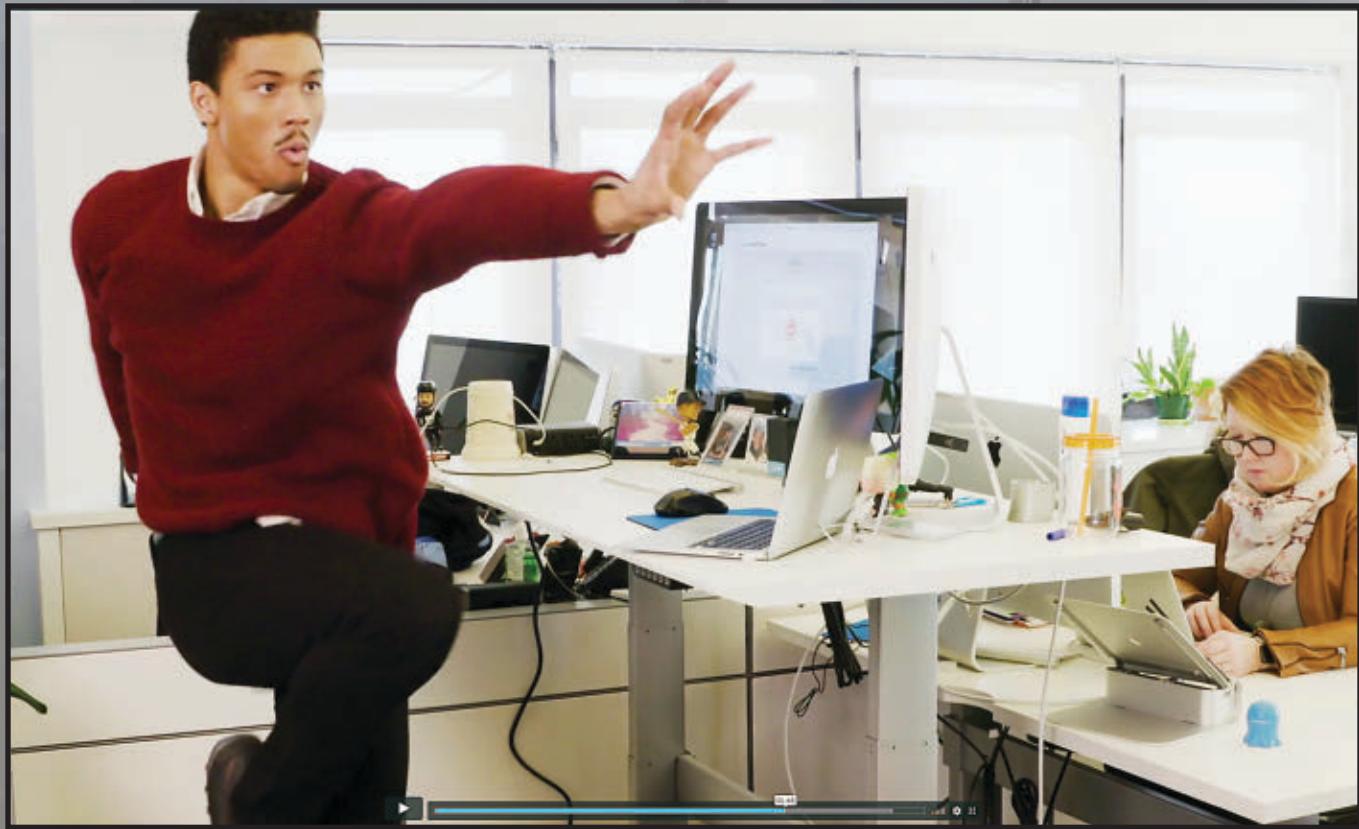
Before discussing the technical details, let's look at what makes an AFP funny. The best AFPs are topical and absurdist.

Topical means it refers to current events or trends. This makes it relevant and "a thinker." Topical would be displaying your website upside-down after a large and highly publicized acquisition by a major Australian competitor. (Australians tell me that kind of joke never gets old.) Doing that to your website otherwise just announces that your Web developers finally read that part of the CSS3 spec.

Secondly, it must be so absurd that it reveals a hidden truth. Absurdist humor is not simply silly for silliness' sake. Absurdism acts as a crucible that burns away all lies to get to the truth.

Stack Overflow's 2017 prank, "Dance Dance Authentication," was both topical and absurdist.³ The prank was a blog post and accompanying demonstration video for Stack Overflow's new (fictional) authentication system. Rather than the usual 2FA (two-factor authentication) system that requires an authenticator app or key fob, this system required users to turn on their webcams and dance their password. This was topical because recent growth in 2FA adoption meant many Internet users were experiencing 2FA for the first time. It was absurdist because it took the added burden and nuisance of 2FA to an extreme. It revealed the truth that badly implemented security sacrifices convenience.

Inspiration for absurdity should come from reality. For example, the Go programming language is an intentionally minimalistic language—



A scene from Stack Overflow's 2017 AFP: Dance Dance Authentication (<https://www.youtube.com/watch?v=VgC4b9K-gYU>).

a reaction against bloated languages such as C++ and Java. It seems like every C++ or Java programmer who learns Go posts to forums demanding dozens of features that are “missing.” This leads to a discussion about why those features are intentionally missing from Go. This discussion seems to happen on a weekly basis. A good AFP for Go would be a blog post announcing that Go 2.0 will include all those “missing” features and, in fact, they have been implemented and are ready for use. The article would then link to the download page for Java.

What Makes an April Fools' Prank Un-Funny?

A prank should not get in the way of business or harm customers. For example, a 2016 Gmail prank called “Drop the Mic” gave users a button that would send a farewell message to someone, then block all email from that person ... forever. There was no “Are you sure?” prompt. As you can guess, this disrupted actual customers trying to do actual business.² Google disabled the prank a few hours later.

An AFP should not mock a particular person (that is just mean) or group of people (that is just hateful). The exception to this is that it is always OK to mock people more powerful than you. Punch up, not down.

► *Punch up:* Mock elite people who don't realize how privileged they are; mock the CEO who bragged he's saving the company money by using his private jet.

► *Don't punch down:* Do not mock the less fortunate—for example, don't mock homeless people or any group of powerless people in society; racist, sexist, or homophobic humor is not funny because it is inherently punching down.

An AFP should be funny to the audience, not just the people who created it. Every year plenty of companies produce AFPs that fall flat because they are inside jokes that everyone in the company finds hiii-larious. That is all well and good, but if the intention was to make customers laugh, it really should not depend on them knowing that Larry in accounting loves *World of Warcraft*.

As with any feature, user acceptance testing should be done with a wide variety of users. Be sure to include some nonusers. You might consider doing user experience testing, but since most companies don't, why start now?

Engineer It Like Any Other Feature

The end-to-end process of creating and launching the prank should be the same as any other feature. It should start with a concept, then have a design and execution plan, launch plan, and operational runbook. Involve product management. Have requirements, specifications, a project schedule, testing, and so on. If it is a big prank, beta testing with users sworn to secrecy may be required.

Like any major feature, the earlier you involve operations, the better. Operations' worst nightmare is to be told that a major feature is being launched tomorrow ... “Would you please set up 10 new servers and find a petabyte of disk space?” April Fools' pranks are no different. They often require extra bandwidth, isolated servers, firewall rules, and other tasks that take days or weeks to complete.

Feature Flags

The prank should be easy to enable and disable. Hide the feature behind a “feature flag.” With the flag off, the feature is in the code but dormant. Enabling the prank in production is a matter of turning the flag on. Disabling it is a simple matter of turning the flag off. Developers can test the feature by enabling the flag in the development and test environments. Some flag systems can automatically be on for certain user segments.

Some companies can launch or disable a feature only by rolling out new code into production. This is bad for many reasons. It is riskier than feature flags: if the release that removes a prank is broken, do you revert to the previous release (with the prank) or the prior release (which may be too old to deploy into production)? Code pushes are difficult to coordinate with PR, blog posts, and so on: they might take minutes or hours, not seconds, like flipping a feature flag. Code pushes require more skill: in many environments, code pushes are done by specific people, who might be asleep. In an emergency you want to empower anyone to shut off the prank. The process should be quick and easy. Lastly, if the prank has overloaded the network, it may also affect the systems that push new code. Meanwhile, a feature “flag flip” is simpler and more likely to just plain work.

The way you structure an AFP project is unusual in that the deadline cannot change. There are three levers available to managers: deadline, budget, and features. If a project is going to be late, management must adjust one of those three. An AFP, however, cannot adjust the deadline and usually has a limited budget. Therefore, it is important to segment the features of the prank. First implement the basic prank, then add “would be nice” features. As you get closer to the deadline, throw away the less important features. When a badly structured prank is late, all features will be 80% done, which means 0% of them can be launched. You blew it. When a well-structured prank is late, 80% of the features are ready to launch, and the customers will be no wiser about the missing 20%. Structuring a project in this way requires skillful planning up front.

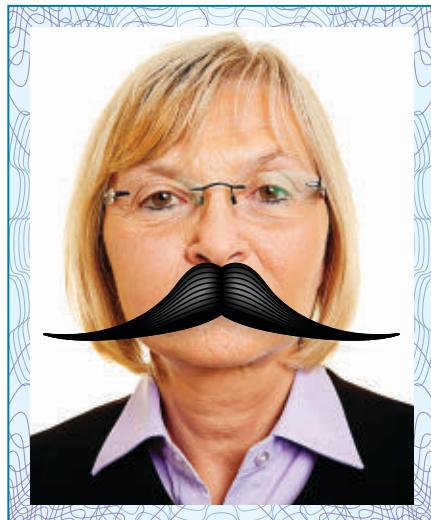
During the prank, plausible deniability is important. Act like it is real, or act like you don’t see it, or act like you were not involved. Do, however, include a link to a page that explains that this is just a joke. They say a joke isn’t funny if you have to explain it; if someone doesn’t realize it is a joke, it can lead to unfunny situations and hurt feelings. This is the Internet, not Mensa.

Perform a project retrospective.⁵ After the prank, sit down with everyone involved and reflect on what went well, what didn’t go well, what should be done the same way next time, and what should have been done differently. Publish this throughout the organization. It not only makes everyone feel included, but it also educates people about how to do better next time. Yes, you may have overloaded the network and created an outage, but if everyone in the organization learned from this experience, your organization is now smarter. Every outage that results in organizational learning is a blessing. If you hide information, the organization stays ignorant.

Case Study: The Mustache Prank

One of the most successful AFPs I was involved with was at a previous employer. Managers had been on a teleconference for an hour brainstorming ideas for an AFP. They wanted one that would be visible only to employees. There is nothing less funny than managers trying to write a joke, so they turned to me. I was a half-manager so they assumed I’d have a half-funny suggestion.

After listening to the ideas they had so far, I was not impressed. They were irrelevant, not topical; silly, not ab-



surdist. Obviously, they did not have the benefit of reading this article.

I thought for a moment. What was the most recently controversy? Well, facial-recognition software was becoming good enough and computationally inexpensive enough that it was making the news and starting a lot of ethical debates.

I blurted out, “Hey, didn’t we just purchase a company that makes facial-recognition software? You’d think a smart bunch of people like that would be able to accurately place mustaches on all the photos in the corporate directory.”

There was a short pause in the conversation. Then one manager said, “We just moved those people into my building. They sit down the hall from me.” Another manager chimed in that he manages the team that runs the corporate directory. Another manages the operations people for it. Another manages the helpdesk most likely to receive any complaints.

Soon, we had a plan.

We started meeting weekly. We wrote a design doc that spelled out how the AFP would work, how we would shut it off after 24 hours, and, most importantly, how individual people could opt out if they complained. A project manager was assigned to coordinate people on three different continents to make it all happen as expected. HR and executive management signed off on the project.

This was long before social media apps were doing this kind of thing, so the primary question we kept getting was, “Is this really possible?”

Was it technically feasible? Yes. It turns out the free software development kit that the company provided included a mustache-placement API. “Mustaching a person” was the demo they used to sell the company.

By the time April 1 rolled around, a new set of photos was prepared and ready to be swapped in. The helpdesk was trained on how to revert individual photos.

The prank was a huge success. Everyone thought it was hilarious, except for one person who complained and opted out.

Afterward, we wrote up a retrospective and thanked everyone involved. In such a highly distributed company, this was the best way to let everyone involved “take a bow.”

Launch It Like It's Hot

If an AFP will have significant resource needs, load testing is important. Everyone knows how to do load testing: simulate thousands of HTTP requests and take measurements. Find and fix the bottlenecks and repeat until you are satisfied.

You also need to plan for the situation where the AFP goes viral and receives 10 times or 100 times more users than you could ever expect. The easy strategy here is simply to plan on disabling the AFP, but it would be disappointing that the reward for success was to turn the feature off.

Fixing such a situation is difficult because normal solutions might take weeks to implement and April Fools' Day lasts only one day. If you fix a problem and relaunch the next day, you have missed the boat.

Facebook is in a similar situation when launching real features because there is a lot of press around a new feature and Facebook needs to "get it right" on the first try. When Facebook was new, growth was slow and bottlenecks could be fixed by simply fixing them at the pace Facebook was growing. By 2008 Facebook had millions of users, and a new feature would go from 0 to millions of users within hours. There would be no time to fix unexpected bottlenecks. A failed launch is highly visible and embarrassing, often becoming front-page news. There is no way, however, to build an isolated system big enough to perform load testing.

To solve this problem, Facebook uses a technique called a "dark launch:" testing a feature by first launching it invisibly. For example, Facebook launched Chat six months early but made it invisible (CSS `display: hidden`). The HTML and JavaScript code was in your browser, but it did not display itself. A certain percentage of users received a signal to send simulated chat messages through the system. The percentage was turned up over time so that developers could spot and fix any performance issues. By the time the feature was made visible (and the test messages were disabled), Facebook's engineers were confident that the launch would not have performance problems. It is suggested that nearly every feature that Facebook will

launch in the next six months is already running in your browser.⁴

Google did something similar before launching IPv6 connectivity; your browser was running invisible JavaScript that tested whether your ISP connection would fail if IPv6 were enabled. Worries were for naught, but the test increased confidence before launch.

Stack Overflow dark launches new ad-serving infrastructure. When launching major features, we first use the system to transmit house ads that are invisible to users. Once performance is verified, we make the advertisements visible. Sadly, we did not use this technique when launching Stack-Egg, but now we know better.

Pranks with Minimal Operational Impact

Technical issues can be avoided with proper testing, but there is a strategy that avoids the issue altogether. Simply create a prank that has no operational impact, or directs the impact elsewhere.

The "Dance Dance Authentication" example is one such prank. The prank was simply a blog post and a link to a YouTube video (<https://www.youtube.com/watch?v=VgC4b9K-gYU>). This does not entirely avoid the issue, but if your success ends up overloading YouTube's network, at least it is not your problem.

You can also simply take an existing feature and create an alternative explanation or history for it. For example, you may have heard of "the teddy bear effect." Many have observed that often the act of asking a question forces you to think out enough details to realize the answer yourself. In Bell Labs folklore there was a researcher known for helping people with research roadblocks. People would go to him for suggestions. By listening, they would come up with the answer themselves. Once, he left on a long vacation and left a teddy bear on his desk with a note that read, "Explain your problem to the bear." Many people found it was equally effective. (Lately, the Internet has started calling this "the rubber duckie effect.")

Suppose you run a question-and-answer website: some users post questions, and other people post answers. Suppose also that the website has a fea-

ture that permits people to write up the answers to their own questions. A very simple but effective AFP would be to rename this feature "teddy bear mode" and write a blog post claiming this to be an entirely new feature, based on the power of a teddy bear's ability to help solve technical issues.

Summary

Successful AFPs require care and planning. Write a design proposal and a project plan. Involve operations early. If this is a technical change to your website, perform load testing, preferably including a "dark launch" or hidden launch test. Hide the prank behind a feature flag rather than requiring a new software release. Perform a retrospective and publish the results widely.

Remember that some of the best AFPs require little or no technical changes at all. For example, one could simply summarize the best practices for launching any new feature but write it under the guise of how to launch an April Fools' prank. That would be hilarious. C

Related articles on queue.acm.org

[Ray Tracing Jell-O Brand Gelatin](#)

Paul S. Heckbert

<http://dl.acm.org/citation.cfm?id=42375>

[The Burning Bag of Dung—and Other Environmental Antipatterns](#)

Phillip Laplante

<http://queue.acm.org/detail.cfm?id=1035617>

[10 Optimizations on Linear Search](#)

Thomas A. Limoncelli

<http://queue.acm.org/detail.cfm?id=2984631>

References

1. Dumke-von der Ehe, B. The making of StackEgg; <http://balphade.de/2015/04/the-making-of-stackegg/>.
2. Kottasova, I. Google's April Fools' prank backfires big time. CNNtech; <http://money.cnn.com/2016/04/01/technology/google-april-fool-prank-backfires/index.html>.
3. Pike, K. Stack Overflow unveils the next steps in computer security. Stack Overflow Blog; <https://stackoverflow.blog/2017/03/30/stack-overflow-unveils-next-steps-computer-security/>.
4. Rossi, C. Pushing millions of lines of code five days a week. Facebook, 2011; <https://www.facebook.com/video/video.php?v=10100259101684977>.
5. Stack Exchange Network Status. Outage postmortem: March 31, 2015; <http://stackstatus.net/post/115305251014/outage-postmortem-march-31-2015>.

Thomas A. Limoncelli is the co-editor of the book, "*The Complete April Fools' Day RFCs*" (<http://www.rfchumor.com>), and is the site reliability engineering manager at Stack Overflow Inc. in New York City. He blogs at EverythingSysadmin.com and tweets @YesThatTom.

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



Perfect should never be the enemy of better.

BY THEO SCHLOSSNAGLE

Monitoring in a DevOps World

THE TITLE OF this article might suggest it is about how you are supposed to be monitoring systems in an organization that is making or has already made the transformation into DevOps. Actually, this is an article to make you think about how computing has changed and how your concept of monitoring perhaps needs re-centering before it even applies to the brave new world of DevOps.

The harsh truth is that this is not just a brave, but also a fast, new world. One of the primary drivers for adopting DevOps is speed—particularly the reduction of risk at speed. An organization has to make many changes to accommodate this. The DevOps community often talks about automation and culture. This makes a lot of sense, as automation is where speed comes

from and every problem can always be rephrased as a people (or communications) problem; automation and culture are key.

That said, the ground has shifted under the monitoring industry. This seismic change has caused existing tools to change and new tools to emerge in the monitoring space, but that alone will not deliver us into the low-risk world of DevOps—not without new and updated thinking. Why? Change.

Monitoring, at its heart, is about observing and determining the behavior of systems, often with an ulterior motive of determining “correctness.” Its purpose is to answer the ever-present question: Are my systems doing what they are supposed to? It’s also worth mentioning that *systems* is a very generic term, and in healthy organizations, systems are seen in a far wider scope than just computers and computing services; they include sales, marketing, and finance, alongside other “business units,” so the business is seen as the complex interdependent system it truly is. That is, good monitoring can help people take a truly *systems view* not only of systems, but also organizations.

Long dead are the systems that age like fine wine. Today’s systems are born in an agile world and remain fluid to accommodate changes in both the supplier and the consumer landscape. A legitimate response to “adapt or die” is “I’ll do DevOps!” This highly dynamic system stands to challenge traditional monitoring paradigms.

The Old World

In a world with “slow” release cycles (often between six and 18 months), making software operational was an interesting challenge. The system deployed at the beginning of a release looked a lot like the same system several months later. It’s not that it was stuck in time, but more that it was branched into a maintenance-only mind-set. With maintenance comes bug fixes and even performance enhancements, but not new features, new systems components, removal of old



systems components, and new features or functions that would fundamentally change the stress on the architecture. Simply put, it is not very fluid.

For monitoring, this lack of fluidity is fantastic. If the system today is the system tomorrow and the exercise that system does today is largely the same tomorrow, then developing a set of expectations around how the system should behave becomes quite natural. From a more pragmatic point of view, the baselines developed by observing the behavior of the system's components will very likely live long, useful lives.

This article is not going to dive into the risks involved with releasing a dramatic set of code changes infrequently, as there are countless stories (anecdotal and otherwise) that state their magnitude and probabilistic certainty. Suffice it to say: there be dragons on that path. This is one of the many reasons that agile, Kanban, and other more re-

sponsive work processes have been so widely adopted. DevOps is the organizational structure that makes the transformation possible.

The New World

So, we are all on board with rapid and fluid business and development processes, and we have continuous "everything" to let us manage risk. The world is wonderful, right? Well, "continuous monitoring" (in this new sense of continuous) doesn't exist, and, besides, the name would be pretty dumb; shouldn't all monitoring have always been continuous?

The big problem here is that the fundamental principles that power monitoring, the very methods that judge if your machine is behaving itself, require an understanding of what good behavior looks like. Whether you are building statistical baselines, using formal models, or just winging it, in order to understand if systems are misbe-

having, you need to know what it looks like when they are behaving.

In this new world, you not only have fluid development processes that can introduce change on a continual basis, you also have adopted a microservices-systems architecture pattern. Microservices simply dictate that the solution to a specific technical problem should be isolated to a network-accessible service with clearly defined interfaces such that the service has freedom. Many developers like this model, as they are given more autonomy in the design of the service, extending to choice of language, database technology, etc. This freedom is very powerful, but its true value lies in decoupling release schedules and maintenance, and allowing for independent higher-level decisions around security, resiliency, and compliance.

This might seem like an odd tangent, but the conflation of these two changes results in something quite

unexpected for the world of monitoring: the system of today neither looks like nor should behave like the system of tomorrow.

An Aside On ML And AI

Many monitoring companies have been struggling to keep up with the nature of ephemeral architecture. Nodes come and go, and architectures dynamically resize from one minute to the next in an attempt to meet growing and shrinking demand. As nodes spin up and subsequently disappear, monitoring solutions must accommodate. While some old monitoring systems struggle with this concept, most modern systems take this type of dynamic systems sizing in stride.

The second and largely unmet challenge is the dynamic nature not of an architecture's size but rather of its design. With microservices-based architectures and multiple agile teams continually releasing software and services, the design of modern architecture is constantly in transition.

A hot topic in monitoring is how to apply ML (machine learning) and AI (artificial intelligence) to the problems at hand, but the current approaches seem to be attempting to solve yesterday's problems and not tomorrow's. AI and ML provide an exceptionally rich new set of techniques to solve problems and will undoubtedly prove instrumental in the monitoring world, but the problems they must tackle are not that of modeling an architecture and learning to guide its operations. The architecture it learns today will have changed by tomorrow, and any guidance will be antiquated. Instead, to make a significant impact, AI and ML approaches need to take a step back and help guide processes and design.

Characteristics of Successful Monitoring

It would be cruel to cast a gloomy shadow on the state of monitoring without providing some tactical advice. Luckily, many people are monitoring their systems exceptionally well. Here is what they have in common:

What is more important than how. The first thing to remember is that all the tools in the world will not help you detect bad behavior if you are looking at the wrong things. Be wary of tools

Today's systems are born in an agile world and remain fluid to accommodate changes in both the supplier and the consumer landscape.

that come with prescribed monitoring for complex assembled systems; rarely are systems in the tech industry assembled and used in the same way at two different organizations. The likely scenario is that the monitoring will seem useless, but in some cases it may provide a false confidence that the systems are functioning well.

When it comes to monitoring the "right thing," always look at your business from the top down. The technical systems the organization operates are only provisioned and operated to meet some stated business goal. Start by monitoring whether that goal is being met. A tongue-in-cheek example: always monitor the payroll system, because if you are not getting paid, what's the point?

Mathematics: It's necessary. Second, embrace mathematics. In modern times, functionality is table stakes; it isn't enough that the system is working, it must be working well. It is a rare day when you have an important monitor that consumes a Boolean value "good" or "bad." Most often, systems are being monitored around delivered performance, so the consumed values (or indicators) are numbers and often latencies (a time representing how long a specific operation took). You are dealing with numbers now, so math is required, like it or not. Basic statistics are a fundamental requirement for both asking and interpreting the answers to questions about the behavior of systems.

As systems grow and the focus turns more to their behavior, data volumes rise. In seven years, Circonus has experienced an increase in data volume of almost seven orders of magnitude. Some people still monitor systems by taking a measurement from them every minute or so, but more and more people are actually observing what their systems are doing. This results in millions or tens of millions of measurements per second on standard servers. People tend not to solve difficult problems unless the answers are valuable. Handling 10 million measurements per second from a single server when you might have thousands of servers might sound like overkill, but people are doing it because the technology exists that makes the cost of finding the answers less than the value of those

answers. People do it because they are able to run better, faster systems and beat the competition. To handle data at that volume, you must also use a capable set of tools. To form intelligent questions around data at this volume, you must embrace mathematics.

As you might imagine, without a set of tools to help you perform fast, accurate, and appropriate mathematical analysis against your observed data, you will be at a considerable disadvantage; luckily, there are myriad choices from Python and R to tools that will help you find more comprehensive solutions from modern monitoring vendors.

Data retention. A third important characteristic of successful monitoring systems is data retention. Monitoring data has often been considered low value and high cost and is often expunged with impudence. Times have changed, and, as with all things computing, the cost of storing data has fallen dramatically. More importantly, DevOps have changed the value of long-term retention of this data. DevOps is a culture of learning. When things go wrong, and they always do, it is critical to have a robust process for interrogating the system and the organization to understand how the failure transpired. This allows processes to be altered to reduce future risk. That's right: learning reduces risk.

At the pace we move, it is undeniable that your organization will develop intelligent questions regarding a failure that were missed immediately after past failures. Those new questions are crucial to the development of your organization, but they become absolutely precious if you can travel back in time and ask those questions about past incidents. This is what data retention in monitoring buys you. The new processes and interrogation methods you learn during your postmortems leading up to this year's cyber-Thursday shopping traffic can now be applied to last year's cyber-Thursday shopping traffic. This often leads to fascinating and valuable learning that, you guessed it, reduces future risk.

Be articulate about what success looks like. The final piece of advice for a successful monitoring system is to be specific about what success looks like. Using a language to articulate what success looks like allows people to win. It is

wholly disheartening to think you've done a good job and met expectations, and then learn the goalposts have moved or that you cannot articulate why you've been successful. The art of the SLI (service-level indicator), SLO (service-level objective), and SLA (service-level agreement) reigns here. Almost every low-level, ad hoc monitor and every high-level executive KPI (key performance indicator) can be articulated in terms of "service level." Understanding the service your business provides and the levels at which you aim to deliver that service is the heart of monitoring.

SLIs are things that you have identified as directly related to the delivery of a service. SLOs are the goals you set for the team responsible for a given SLI. SLAs are SLOs with consequences, often financial. Though a slight oversimplification, think about it like this: What is important? What should it look like? What should I promise? For this, a good understanding of histograms can help.

From RUM to RSM

Monitoring in the Web world moved long ago from the slow, synthetic ping of a website to recording and analyzing every interaction with every user; synthetic monitoring of the web gave way to RUM (real user monitoring) at the turn of the century and no one looked back. As we build more smaller, decoupled services, we move into a realm of being responsible for servicing other small systems—these are what engineering SLOs are usually built around.

The days of average latency for an API request or a database interaction or a disk operation (or even a syscall!) are disappearing. RSM (real systems monitoring) is coming, and we will, just as with RUM, be recording and analyzing systems-level interactions—every one of them. Over the last decade increased systems observability (such as the widely adopted DTrace and Linux's eBPF) and improvements in time-series databases (such as the first-class histogram storage in Circonus's IRONdb) have made it possible to deliver RSM. (For a detailed look at why histogram storage of data is different and, more importantly, relevant, see the review by Baron Schwartz, "Why Percentiles Don't Work the Way You Think." <https://www.vividecortex.com/>

blog/why-percentiles-dont-work-the-way-you-think.)

RSM allows you to look at actual system behavior comprehensively, accounting for the whole distribution of observed performance instead of the synthetically induced measurements that consistently misrepresent the experience of using the system.

The transition from synthetic web monitoring to RUM was seismic; expect nothing less from the impending transition to RSM.

Don't Delay

Today, with architectures dynamically shifting in size by the minute or hour and shifting in design by the day or the week, we need to step back and remember that monitoring is about understanding the behavior of systems, and that systems need not be limited to computers and software. A business is a complex system itself, including decoupled but connected subsystems of sales, marketing, engineering, finance, and so on. Monitoring can be applied to all of these systems to measure important indicators and detect changes in overall systems behavior.

Monitoring can seem quite overwhelming. The most important thing to remember is that *perfect* should never be the enemy of *better*. DevOps enables highly iterative improvement within organizations. If you have no monitoring, get something; get *anything*. Something is better than nothing, and if you have embraced DevOps, you have already signed up for making it better over time. □

Related articles on queue.acm.org

Black Box Debugging

James A. Whittaker, Herbert H. Thompson
<https://queue.acm.org/detail.cfm?id=966807>

Statistics for Engineers

Heinrich Hartmann
<https://queue.acm.org/detail.cfm?id=2903468>

Time, but Faster

Theo Schlossnagle
<https://queue.acm.org/detail.cfm?id=3036398>

Theo Schlossnagle is a software engineer and serial entrepreneur. He currently leads Circonus, where he focuses on building distributed systems and scale to help people analyze the behavior of their distributed systems at scale over time.

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

contributed articles

DOI:10.1145/3127323

As the software industry enters the era of language-oriented programming, it needs programmable programming languages.

BY MATTHIAS FELLEISEN, ROBERT BRUCE FINDLER,
MATTHEW FLATT, SHIRIRAM KRISHNAMURTHI, ELI BARZILAY,
JAY MCCARTHY, AND SAM TOBIN-HOCHSTADT

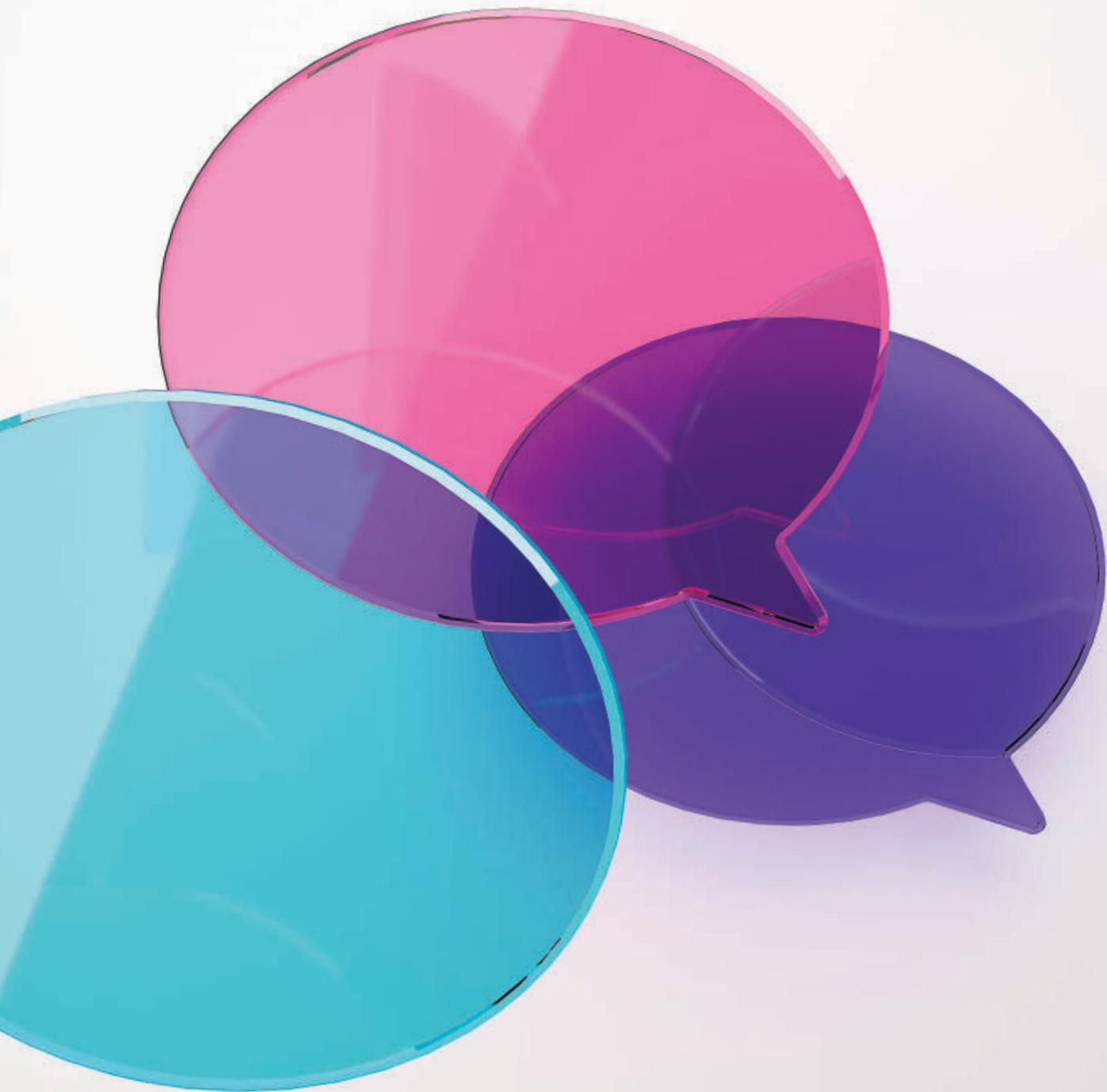
A Programmable Programming Language

IN THE IDEAL world, software developers would analyze each problem in the language of its domain and then articulate solutions in matching terms. They could thus easily communicate with domain experts and separate problem-specific ideas from the details of general-purpose languages and specific program design decisions.

In the real world, however, programmers use a mainstream programming language someone else picked for them. To address this conflict, they resort to—and on occasion build their own—domain-specific languages embedded in the chosen language (embedded domain-specific languages, or eDSLs). For example, JavaScript programmers employ jQuery for interacting with the Document Object Model and React for dealing with events and concurrency.

» key insights

- Language-oriented programming is an emerging software-development paradigm likely to revolutionize the way people build software.
- It elevates “language” itself to a software building block, with the same status as objects, modules, and components.
- As with other paradigms, language orientation thrives when the base language supports it directly; the Racket project has worked on support for language-oriented programming for 20 years, providing a platform for exploring this exciting new development in depth.



As developers solve their problems in appropriate eDSLs, they compose these solutions into one system; that is, they effectively write multilingual software in a common host language.^a

Sadly, multilingual eDSL programming is done today on an ad hoc basis

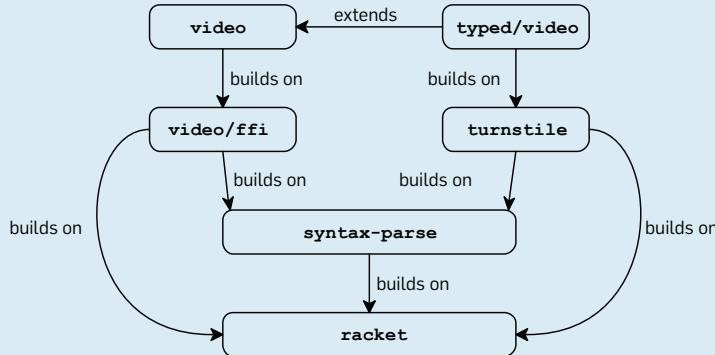
and is rather cumbersome. To create and deploy a language, programmers usually must step outside the chosen language to set up configuration files and run compilation tools and link-in the resulting object-code files. Worse, the host languages fail to support the proper and sound integration of components in different eDSLs. Moreover, most available integrated development environments (IDEs) do not even understand eDSLs or perceive the presence of code written in eDSLs.

The goal of the Racket project is to explore this emerging idea of lan-

guage-oriented programming, or LOP, at two different levels. At the practical level, the goal is to build a programming language that enables language-oriented software design. This language must facilitate easy creation of eDSLs, immediate development of components in these newly created languages, and integration of components in distinct eDSLs; Racket is available at <http://racket-lang.org/>

At the conceptual level, the case for LOP is analogous to the ones for object-oriented programming and for concurrency-oriented programming.³ The

^a The numerous language-like libraries in scripting languages (such as JavaScript, Python, and Ruby), books (such as Fowler and Parson),²⁰ and websites (such as Federico Tomassetti's, <https://tomassetti.me/resources-create-programming-languages/>) are evidence of the desire by programmers to use and develop eDSLs.

Figure 1. Small language-oriented programming example.**Figure 2. A plain Racket module.**

```

#lang racket/base
(provide
  ;; type MaxPath = [Listof Edge]
  ;; Natural -> MaxPath
  walk-simplex)

(require "constraints" graph)

;; Natural -> MaxPath
(define (walk-simplex timing)
  ... (maximizer #:x 2)...)
  
```

demo

Figure 3. A module for describing a simplex shape.

```

#lang simplex
;; implicitly provides synthesized function maximizer:
;; #:x Real -> Real
;; #:y Real -> Real

#:variables x y

3 * x + 5 * y <= 10
3 * x - 5 * y <= 20
  
```

constraints

Figure 4. Lambda, redefined.

```

01 #lang racket
02
03 (provide (rename-out [new-lambda lambda]))
04
05 (require (for-syntax syntax/parse))
06 ...
07 ; Syntax -> Syntax
08 (define-syntax (new-lambda stx)
09   (syntax-parse stx
10     [((new-lambda (x:id (~literal ::) predicate:id) body:expr)
11       (syntax
12         (lambda (x)
13           (unless (predicate x)
14             (define name (object-name predicate))
15             (error 'lambda "a expected, given: ~e" name x)))
16       body))]))
17 ...
  
```

new-lam

former arose from making the creation and manipulation of objects syntactically simple and dynamically cheap, the latter from Erlang's inexpensive process creation and message passing. Both innovations enabled new ways to develop software and triggered research projects. The question is how our discipline will realize LOP and how it will affect the world of software.

Our decision to develop a new language—Racket—is partly an historical artifact and partly due to our desire to free ourselves from any unnecessary constraints of industrial mainstream languages as we investigate LOP. The next section spells out how Racket got started, how we honed in on LOP, and what the idea of LOP implies.

Principles of Racket

The Racket project dates to January 1995 when we started it as a language for experimenting with pedagogic programming languages.¹⁵ Working on them quickly taught us that a language itself is a problem-solving tool. We soon found ourselves developing different languages for different parts of the project: a (meta-) language for expressing many pedagogic languages, another for specializing the DrRacket IDE,¹⁵ and a third for managing configurations. In the end, the software was a multilingual system, as outlined earlier.

Racket's guiding principle reflects the insight we gained: empower programmers to create new programming languages easily and add them with a friction-free process to a codebase. By “language,” we mean a new syntax, a static semantics, and a dynamic semantics that usually maps the new syntax to elements of the host language and possibly external languages via a foreign-function interface (FFI). For a concrete example, see Figure 1 for a diagram of the architecture of a recently developed pair of scripting languages for video editing^{2,b} designed to assist people who turn recordings of conference presentations into YouTube videos and channels. Most of that work is

^b The video language, including an overview of the implementation, is available as a use-case artifact at <https://www2.ccs.neu.edu/racket/pubs/#icfp17-acf>

repetitive—adding preludes and post-ludes, concatenating playlists, and superimposing audio—with few steps demanding manual intervention. This task calls for a domain-specific scripting language; video is a declarative eDSL that meets this need.

The typed/video language adds a type system to video. Clearly, the domain of type systems comes with its own language of expertise, and typed/video's implementation thus uses turnstile,⁶ an eDSL created for expressing type systems. Likewise, the implementation of video's rendering facility calls for bindings to a multimedia framework. Ours separates the binding definitions from the repetitive details of FFI calls, yielding two parts: an eDSL for multimedia FFIs, dubbed video/ffi, and a single program in the eDSL. Finally, in support of creating all these eDSLs, Racket comes with the syntax parse eDSL,⁷ which targets eDSL creation.

The LOP principle implies two subsidiary guidelines:

Enable creators of a language to enforce its invariants. A programming language is an abstraction, and abstractions are about integrity. Java, for example, comes with memory safety and type soundness. When a program consists of pieces in different languages, values flow from one context into another and need protection from operations that might violate their integrity, as we discuss later; and

Turn extra-linguistic mechanisms into linguistic constructs. A LOP programmer who resorts to extra-linguistic mechanisms effectively acknowledges that the chosen language lacks expressive power.^{13,c} The numerous external languages required to deal with Java projects—a configuration language, a project description language, and a makefile language—represent symptoms of this problem. We treat such gaps as challenges later in the article.

They have been developed in a feedback loop that includes DrRacket¹⁵ plus typed,³⁶ lazy,⁴ and pedagogical languages.¹⁵

c Like many programming-language researchers, we subscribe to a weak form of the Sapir-Whorf hypothesis; see <http://docs.racket-lang.org/algol60/> and <https://www.hashcollision.org/brainfudge/> showing how Racket copes with obscure syntax.

Most notably, Racket eliminates the hard boundary between library and language, overcoming a seemingly intractable conflict.

Libraries and Languages Reconciled

Racket is an heir of Lisp and Scheme. Unlike these ancestors, however, Racket emphasizes functional over imperative programming without enforcing an ideology. Racket is agnostic when it comes to surface syntax, accommodating even conventional variants (such as Algol 60).^d Like many languages, Racket comes with “batteries included.”

Most notably, Racket eliminates the hard boundary between library and language, overcoming a seemingly intractable conflict. In practice, this means new linguistic constructs are as seamlessly imported as functions and classes from libraries and packages. For example, Racket's class system and for loops are imports from plain libraries, yet most programmers use these constructs without ever noticing their nature as user-defined concepts.

Racket's key innovation is a modular syntax system,^{17,26} an improvement over Scheme's macro system,^{11,24,25} which in turn improved on Lisp's tree-transformation system. A Racket module provides such services as functions, classes, and linguistic constructs. To implement them, a module may require the services of other modules. In this world of modules, creating a new language means simply creating a module that provides the services for a language. Such a module may subtract linguistic constructs from a base language, reinterpret others, and add a few new ones. A language is rarely built from scratch.

Like Unix shell scripts, which specify their dialect on the first line, every Racket module specifies its language on the first line, too. This language specification refers to a file that contains a language-defining module. Creating this file is all it takes to install a language built in Racket. Practically speaking, a programmer may develop a language in one tab of the IDE, while another tab may be a module written in the language of the first. Without ever leaving the IDE to run compilers, linkers, or other tools, the developer can modify the language implementation in the first tab and immediately experience the modification in the second;

d See <http://docs.racket-lang.org/algol60/>, as well as <https://www.hashcollision.org/brainfudge/>, which shows how Racket copes with obscure syntax.

that is, language development is a friction-free process in Racket.

In the world of shell scripts, the first-line convention eventually opened the door to a slew of alternatives to shells, including Perl, Python, and Ruby. The Racket world today reflects a similar phenomenon, with language libraries proliferating within its ecosystem: racket/base, the Racket core language; racket, the “batteries included” variant; and typed/racket, a typed variant. Some lesser-known examples are datalog and a web-server language.^{27,30} When precision is needed, we use the lowercase name of the language in typewriter font; otherwise we use just “Racket.”

Figure 2 is an illustrative module. Its first line—pronounced “hash lang racket base”—says it is written in racket/base. The module provides a single function, walk-simplex. The accompanying line comments—introduced with semicolons—informally state a type definition and a function signature in terms of this type definition; later, we show how developers can use typed/racket to replace such comments with statically checked types, as in Figure 5. To implement this function, the module imports functionality from the constraints module outlined in Figure 3. The last three lines of Figure 2 sketch the definition of the walk-simplex function, which refers to the maximizer function imported from constraints.

The “constraints” module in Figure 3 expresses the implementation of its only service in a domain-specific language because it deals with simplexes, which are naturally expressed through a system of inequalities. The module’s simplex language inherits the line-comment syntax from racket/base but uses infix syntax otherwise. As the comments state, the module exports a single function, maximizer, which consumes two optional keyword parameters. When called as (maximizer #:x n), as in Figure 2, it produces the maximal y value of the system of constraints. As in the lower half of Figure 3, these constraints are specified with conventional syntax.

In support of this kind of programming, Racket’s modular syntax system benefits from several key innovations. A particularly illustrative one is the ability to incrementally redefine the meaning of existing language con-

**In general,
cooperating
multilingual
components
must respect
the invariants
established by
each participating
language.**

structs via the module system. It allows eDSL creators to ease their users into a new language by reusing familiar syntax, but reinterpreted.

Consider lambda expressions, for example. Suppose a developer wishes to equip a scripting language (such as video) with functions that check whether their arguments satisfy specified predicates. Figure 4 shows the basic idea:

line 01 The module uses the racket language.

line 03 It exports a defined compile-time function, new-lambda, under the name lambda, which is overlined in the code to mark its origin as this module.

line 05 Here, the module imports tools from a library for creating robust compile-time functions conveniently.⁷

line 07 The comment says a function on syntax trees follows.

line 08 While (define (f x) . . .) introduces an ordinary function f of x, (define-syntax (c stx) . . .) creates the compile-time function c with a single argument, stx.

line 09 As with many functional languages, Racket comes with pattern-matching constructs. This one uses syntax-parse from the library mentioned earlier. Its first piece specifies the to-be-matched tree (stx); the remainder specifies a series of pattern-responses clauses.

line 10 This pattern matches any syntax tree with first token as new-lambda followed by a parameter specification and a body. The annotation :id demands that the pattern variables x and predicate match only identifiers in the respective positions. Likewise, :expr allows only expressions to match the body pattern variable.

line 11 A compile-time function synthesizes new trees with syntax.

line 12 The generated syntax tree is a lambda expression. Specifically, the function generates an expression that uses lambda. The underline in the code marks its origin as the ambient language, here racket.

other lines Wherever the syntax system encounters the pattern variables x, predicate, and body, it inserts the respective subtrees that match x, predicate, and body.

When another module uses “new-lam” as its language, the compiler

elaborates the surface syntax into the core language like this

```
(lambda (x :: integer?) (+ x 1))
-elaborates to→
Lambda (x :: integer?) (+ x 1)
-elaborates to→
(new-lambda (x :: integer?) (+ x 1))
-elaborates to→
(lambda (x)
  (unless (integer? x)
    <elided error reporting>
  (+ x 1)))
```

The first elaboration step resolves `lambda` to its imported meaning,¹⁸ or `Lambda`. The second reverses the “rename on export” instruction. Finally, the `new-lambda` compile-time function translates the given syntax tree into a `racket` function.

In essence, Figure 4 implements a simplistic precondition system for one-argument functions. Next, the language developer might wish to introduce multi-argument `lambda` expressions, add a position for specifying the post-condition, or make the annotations optional. Naturally, the compile-time functions could then be modified to check some or all of these annotations statically, eventually resulting in a language that resembles `typed/racket`.

Sound Cooperation Between Languages

A LOP-based software system consists of multiple cooperating components, each written in domain-specific languages. Cooperation means the components exchange values, while “multiple languages” implies these values are created in distinct languages. In this setting, things can easily go wrong, as demonstrated in Figure 5 with a toy

scenario. On the left, a module written in `typed/racket` exports a numeric differentiation function. On the right, a module written in `racket` imports this function and applies it in three different ways, all illegal. If such illegal uses of the function were to go undiscovered, developers would not be able to rely on type information for designing functions or for debugging, nor could compilers rely on them for optimizations. In general, cooperating multilingual components must respect the invariants established by each participating language.

In the real world, programming languages satisfy a spectrum of guarantees about invariants. For example, C++ is unsound. A running C++ program may apply any operation to any bit pattern and, as long as the hardware does not object, program execution continues. The program may even terminate “normally,” printing all kinds of output after the misinterpretation of the bits. In contrast, Java does not allow the misrepresentation of bits but is only somewhat more sound than C++.¹⁹ ML improves on Java again and is completely sound, with no value ever manipulated by an inappropriate operation.

`Racket` aims to mirror this spectrum of soundness at two levels: language implementation itself and cooperation between two components written in different embedded languages. First consider the soundness of languages. As the literature on domain-specific languages suggests,²⁰ such languages normally evolve in a particular manner, as is true for the `Racket` world, as in Figure 6. A first implementation is often a thin veneer over an efficient C-level API. `Racket` developers

create such a veneer with a foreign interface that allows parenthesized C-level programming.⁵ Programmers can refer to a C library, import functions and data structures, and wrap these imports in `Racket` values. Figure 7 illustrates the idea with a sketch of a module; `video`’s initial implementation consisted of just such a set of bindings to a video-rendering framework. When a `racket/base` module imports the `ffi/unsafe` library, the language of the module is unsound.

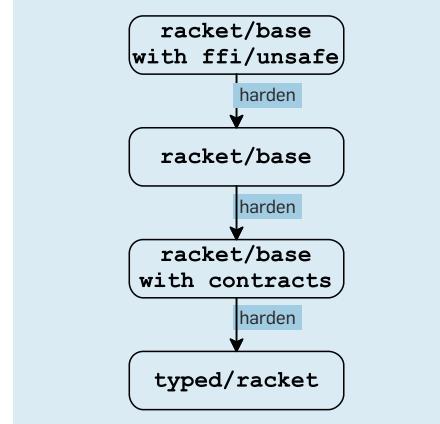
A language developer who starts with an unsound eDSL is likely to make it sound as the immediate next step. To this end, the language is equipped with runtime checks similar to those found in dynamically typed scripting languages to prevent the flow of bad values to unsound primitives. Unfortunately, such protection is ad hoc, and, unless developers are hypersensitive, the error messages may originate from inside the library, thus blaming some `racket/base` primitive operation for the error. To address this problem, `Racket` comes with higher-order contracts¹⁶ with which a language developer might uniformly protect the API of a library from bad values. For example, the `video/ffi` language provides language constructs for making the bindings to the video-rendering framework safe. In addition to plain logical assertions, `Racket`’s developers are also experimenting with contracts for checking protocols, especially temporal ones.⁹ The built-in blame mechanism of the contract library ensures sound blame assignment.¹⁰

Finally, a language developer may wish to check some logical invariants before the programs run. Checking simple types is one example, though

Figure 5. Protecting invariants.

<pre>#lang typed/racket</pre>	TR	<pre>#lang racket</pre>	RR
<pre>(provide diff)</pre>		<pre>(require "TR.rkt")</pre>	
<pre>(: diff ((Real -> Real) -> (Real -> Real))) (define (diff f) (lambda (x) (define lo (f (- x eps))) (define hi (f (+ x eps))) (/ (- hi lo) (* 2 eps))))</pre>		<pre>;; scenario 1 (diff 1) ;; scenario 2 (define (f-bool x) #true) (diff f-bool) ;; scenario 3 (define (f-char x) (string x x)) (diff f-str)</pre>	

Figure 6. Hardening a module.



other forms of static checking are also possible. The typed/video language illustrates this point with a type system that checks the input and output types of functions that may include numeric constraints on the integer arguments; as a result, no script can possibly render a video of negative length. Likewise, typed/racket is a typed variant of (most of) racket.

Now consider the soundness of cooperating languages. It is again up to the language developer to anticipate how programs in this language interact with others. For example, the creator of typed/video provides no protection for its programs. In contrast, the cre-

ators of typed/racket intended the language to be used in a multilingual context; typed/racket thus compiles the types of exported functions into the higher-order contracts mentioned. When, for example, an exported function must always be applied to integer values, the generated contract inserts a check that ensures the “integerness” of the argument at every application site for this function; there is no need to insert such a check for the function’s return points because the function is statically type checked. For a function that consumes an integer-valued function, the contract must ensure the function argument always returns an integer. In

general, a contract wraps exported values with a proxy³¹ that controls access to the value. The idea is due to Matthews and Findler,²⁹ while Tobin-Hochstadt’s and Felleisen’s Blame Theorem³⁵ showed that if something goes wrong with such a mixed system, the runtime exception points to two faulty components and their boundary as the source of the problem.¹⁰ In general, Racket supplies a range of protection mechanisms, and a language creator can use them to implement a range of soundness guarantees for cooperating eDSLs.

Universality vs. Expressiveness

Just because a general-purpose language can compute all partial-recursive functions, programmers cannot necessarily express all their ideas about programs in this language.¹³ This point is best illustrated through an example. So, imagine the challenge of building an IDE for a new programming language in the very same language. Like any modern IDE, it is supposed to enable users to compile and run their code. If the code goes into an infinite loop, the user must be able to terminate it with a simple mouse click. To implement this capability in a natural^e manner, the language must internalize the idea of a controllable process, a thread. If it does not internalize such a notion, the implementer of the IDE must step outside the language and somehow re-use processes from the underlying operating system.

For a programming language researcher, “stepping outside the language” signals failure. Or, as Ingalls²¹ said, “[an] operating system is a collection of things that don’t fit into a language[; t]here shouldn’t be one.” We, Racket creators, have sought to identify services Racket borrows from the surrounding operating system and assimilate them into the language itself.¹⁹ Here are three sample constructs for which programmers used to step outside of Racket but no longer need to:

Sandboxes. That restrict access to resources;

Inspectors. That control reflective capabilities; and

^e An alternative is to rewrite the entire program before handing it to the given compiler, exactly what distinguishes “expressiveness” from “universality.”

Figure 7. A Racket module using the foreign-function interface.

```
#lang racket/base
[ffi]
(provide
  ( ; [Vectorof [Vectorof Real]] -> [Vectorof Real]
  simplex)

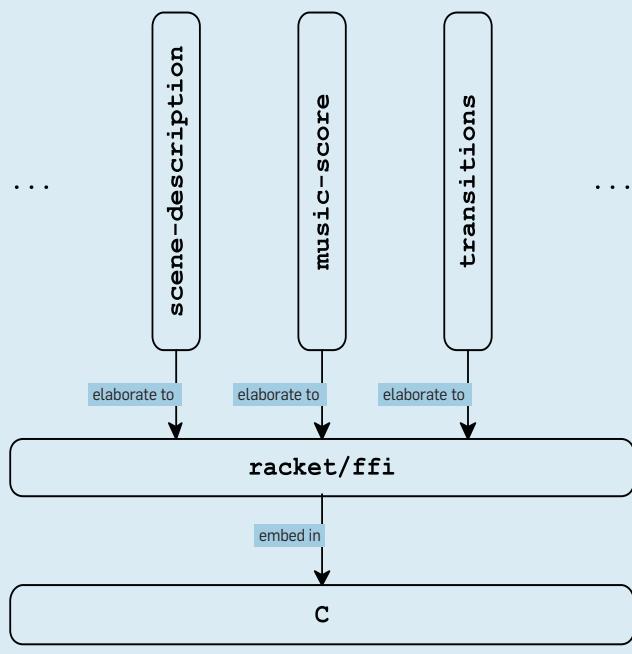
(require ffi/unsafe)

(define (simplex M)
  ... (ffi-simplex-set ...) ...)

(define lib-simplex (ffi-lib "./coin-Clp/lib/libClp"))

(define ffi-simplex-set
  (get-ffi-obj "simplex" lib-simplex (_fun _bytes -> _void)))
```

Figure 8. A sketch of an industrial example of language-oriented programming.



Custodians. That manage resources (such as threads and sockets).

To understand how inclusion of such services helps language designers, consider a 2014 example, the *shill* language.³² Roughly speaking, *shill* is a secure scripting language in Racket's ecosystem. With *shill*, a developer articulates fine-grain security and resource policies—along with, say, what files a function may access or what binaries the script may run—and the language ensures these constraints are satisfied. To make this concrete, consider a homework server to which students can submit their programs. The instructor might wish to run an auto-grade process for all submissions. Using a *shill* script, the homework server can execute student programs that cannot successfully attack the server, poke around in the file system for solutions, or access external connections to steal other students' solutions. Naturally, *shill*'s implementation makes extensive use of Racket's means of running code in sandboxes and harvesting resources via custodians.

State of Affairs

The preceding sections explained how Racket enables programmers to do the following:

Create languages. Create by way of linguistic reuse for specific tasks and aspects of a problem;

Equip with soundness. Equip a language with almost any conventional level of soundness, as found in ordinary language implementations; and

Exploit services. Exploit a variety of internalized operating system services for constructing runtime libraries for these embedded languages.

What makes such language-oriented programming work is “incrementality,” or the ability to develop languages in small pieces, step by step. If conventional syntax is not a concern, developers can create new languages from old ones, one construct at a time. Likewise, they do not have to deliver a sound and secure product all at once; they can thus create a new language as a wrapper around, say, an existing C-level library, gradually tease out more of the language from the interface, and make the language as sound or secure as time permits or a growing user base demands.

Racket borrows from the surrounding operating system and assimilates such extra-linguistic mechanisms into the language itself.

Moreover, the entire process takes place within the Racket ecosystem. A developer creates a language as a Racket module and installs it by “importing” it into another module. This tight coupling has two implications: the development tools of the ecosystem can be used for creating language modules and their clients; and the language becomes available for creating more languages. Large projects often employ a tower involving a few dozen languages, all helping manage the daunting complexity in modern software systems.

Sony's *Naughty Dog* game studio has created just such a large project, actually a framework for creating projects. Roughly speaking, Sony's Racket-based architecture provides languages for describing scenes, transitions between scenes, scores for scenes, and more. Domain specialists use the languages to describe aspects of the game. The Racket implementation composes these domain-specific programs, then compiles them into dynamically linked libraries for a C-based game engine; Figure 8 sketches the arrangement graphically.

Racket's approach to language-oriented programming is by no means perfect. To start with, recognizing when a library should become a language requires a discriminating judgment call. The next steps require good choices in terms of linguistic constructs, syntax, and runtime primitives.

As for concrete syntax, Racket currently has strong support for typical, incremental Lisp-style syntax development, including traditional support for conventional syntax, or generating lexers and parsers. While traditional parsing introduces the natural separation between surface syntax and meaning mentioned earlier, it also means the development process is no longer incremental. The proper solution would be to inject Racket ideas into a context where conventional syntax is the default.^f

^f Language workbenches (such as Spooftax²²) deal with conventional syntax for DSLs but do not support the incremental modification of existing languages. A 2015 report¹² suggests, however, these tool chains are also converging toward the idea of language creation as language modification. We conjecture that, given sufficient time, development of Racket and language workbenches will converge on similar designs.

As for static checking, Racket forces language designers to develop such checkers wholesale, not incrementally. The type checker for typed/racket looks like, for example, the type checker for any conventionally typed language; it is a complete recursive-descent algorithm that traverses the module's representation and algebraically checks types. What Racket developers really want is a way to attach type-checking rules to linguistic constructs, so such algorithms can be synthesized as needed.

Chang et al.⁶ probably took a first step toward a solution for this problem and have thus far demonstrated how their approach can equip a DSL with any structural type system in an incremental and modular manner. A fully general solution must also cope with substructural type systems (such as the Rust programming language) and static program analyses (such as those found in most compilers).

As for dynamic checking, Racket suffers from two notable limitations: On one hand, it provides the building blocks for making language cooperation sound, but developers must create the necessary soundness harnesses on an ad hoc basis. To facilitate the composition of components in different languages, Racket developers need both a theoretical framework and abstractions for the partial automation of this task. On the other hand, the available spectrum of soundness mechanisms lacks power at both ends, and how to integrate these powers seamlessly is unclear. To achieve full control over its context, Racket probably needs access to assembly languages on all possible platforms, from hardware to browsers. To realize the full power of types, typed/racket will have to be equipped with dependent types. For example, when a Racket program uses vectors, its corresponding typed variant type-checks what goes into them and what comes out, but like ML or Haskell, indexing is left to a (contractual) check in the runtime system. Tobin-Hochstadt and his Typed Racket group are working on first steps in this direction, focusing on numeric constraints,²³ similar to Xi's and Pfenning's research.³⁷

As for security, the Racket project is still looking for a significant break-

To achieve full control over its context, Racket probably needs access to assembly languages on all possible platforms, from hardware to browsers.

through. While the shill team was able to construct the language inside the Racket ecosystem, its work exposed serious gaps between Racket's principle of language-oriented programming and its approach to enforcing security policies. It thus had to alter many of Racket's security mechanisms and invent new ones. Racket must clearly make this step much easier, meaning more research is needed to turn security into an integral part of language creation.

Finally, LOP also poses brand-new challenges for tool builders. An IDE typically provides tools for a single programming language or a family of related languages, including debuggers, tracers, and profilers. Good tools communicate with developers in terms of the source language. Due to its very nature, LOP calls for customization of such tools to many languages, along with their abstractions and invariants. We have partially succeeded in building a tool for debugging programs in the syntax language,⁸ have the foundations of a debugging framework,²⁸ and started to explore how to infer scoping rules and high-level semantics for newly introduced, language-level abstractions.^{33,34} Customizing these tools automatically to newly created (combinations of) languages remains an open challenge.

Conclusion

Programming language research is short of its ultimate goal—provide software developers tools for formulating solutions in the languages of problem domains. Racket is one attempt to continue the search for proper linguistic abstractions. While it has achieved remarkable success in this direction, it also shows that programming-language research has many problems to address before the vision of language-oriented programming becomes reality.

Acknowledgments

We thank Claire Alvis, Robert Cartwright, Ryan Culpepper, John Clements, Stephen Chang, Richard Cobbe, Greg Cooper, Christos Dimoulas, Bruce Duba, Carl Eastlund, Burke Fetscher, Cormac Flanagan, Kathi Fisler, Dan Friedman, Tony Garnock

Jones, Paul Graunke, Dan Grossman, Kathy Gray, Casey Klein, Eugene Kohlbecker, Guillaume Marceau, Jacob Matthews, Scott Owens, Greg Pettyjohn, Jon Rafkind, Vincent St-Amour, Paul Steckler, Stevie Strickland, James Swaine, Asumu Takikawa, Kevin Tew, Neil Toronto, and Adam Wick for their contributions.

A preliminary version of this article appeared in the *Proceedings of the First Summit on Advances in Programming Languages* conference in 2015.¹⁴ In addition to its reviewers, Sam Caldwell, Eduardo Cavazos, John Clements, Byron Davies, Ben Greenman, Greg Hendershott, Manos Renieris, Marc Smith, Vincent St-Amour, and Asumu Takikawa suggested improvements to the presentation of this material. The anonymous *Communications* reviewers challenged several aspects of our original submission and thus forced us to greatly improve the exposition.

Since the mid-1990s, this work has been generously supported by our host institutions—Rice University, University of Utah, Brown University, University of Chicago, Northeastern University, Northwestern University, Brigham Young University, University of Massachusetts Lowell, and Indiana University—as well as a number of funding agencies, foundations, and companies, including the Air Force Office of Scientific Research, Cisco Systems Inc., the Center for Occupational Research and Development, the Defense Advanced Research Projects Agency, the U.S. Department of Education’s Fund for the Improvement of Postsecondary Education, the ExxonMobil Foundation, Microsoft, the Mozilla Foundation, the National Science Foundation, and the Texas Advanced Technology Program. □

References

1. Amin, N. and Tate, R. Java and Scala’s type systems are unsound: The existential crisis of null pointers. In *Proceedings of ACM SIGPLAN conference on Object-Oriented Programming Systems, Languages & Applications*, 2016, 838–848.
2. Andersen, L., Chang, S., and Felleisen, M. Super 8 languages for making movies. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, 2017, 1–29.
3. Armstrong, J. Concurrency-oriented programming. In *Fruhjahrsgespräch der German Unix User Group*, 2003; <http://guug.de/veranstaltungen/ffg2003/papers/>
4. Barzilay, E. and Clements, J. Laziness without all the hard work. In *Proceedings of the Workshop on Functional and Declarative Programming in Education*, 2005, 9–13.
5. Barzilay, E. and Orlofsky, D. Foreign interface for PLT Scheme. In *Proceedings of the Ninth ACM SIGPLAN Workshop on Scheme and Functional Programming*, 2004, 63–74.
6. Chang, S., Knauth, A., and Greenman, B. Type systems as macros. In *Proceedings of the 44th ACM SIGPLAN Principles of Programming Languages*, 2017, 694–705.
7. Culpepper, R. Fortifying macros. *Journal of Functional Programming* 22, 4–5 (Aug. 2012), 439–476.
8. Culpepper, R. and Felleisen, M. Debugging macros. *Science of Computer Programming* 75, 7 (July 2010), 496–515.
9. Dimoulas, C., New, M., Findler, R., and Felleisen, M. Oh Lord, please don’t let contracts be misunderstood. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, 2016, 117–131.
10. Dimoulas, C., Tobin-Hochstadt, S., and Felleisen, M. Complete monitors for behavioral contracts. In *Proceedings of the European Symposium on Programming*, 2012, 214–233.
11. Dybvig, R., Hieb, R., and Bruggeman, C. Syntactic abstraction in Scheme. *Lisp and Symbolic Computation* 5, 4 (Dec. 1993), 295–326.
12. Erdweg, S., van der Storm, T., Vlter, M., Tratt, L., Bosman, R., Cook, W.R., Gerritsen, A., Hulshout, A., Kelly, S., Loh, A., Konat, G., Molina, P.J., Palatin, M., Pohjonen, R., Schindler, E., Schindler, K., Solmi, R., Vergu, V., Visser, E., van der Vlist, K., Wachsmuth, G., and van der Woning, J. Evaluating and comparing language workbenches: Existing results and benchmarks for the future. *Computer Languages, Systems and Structures* 44, Part A (Dec. 2015), 24–47.
13. Felleisen, M. On the expressive power of programming languages. *Science of Computer Programming* 17, 1–3 (Dec. 1991), 35–75.
14. Felleisen, M., Findler, R.B., Flatt, M., Krishnamurthi, S., Barzilay, E., McCarthy, J., and Tobin-Hochstadt, S. The Racket Manifesto. In *Proceedings of the First Summit on Advances in Programming Languages*, T. Ball, R. Bodik, S. Krishnamurthi, B.S. Lerner, and G. Morrisett, Eds. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2015, 113–128.
15. Findler, R., Clements, J., Flanagan, C., Flatt, M., Krishnamurthi, S., Steckler, P., and Felleisen, M. DrScheme: A programming environment for Scheme. *Journal of Functional Programming* 12, 2 (Mar. 2002), 159–182.
16. Findler, R.B. and Felleisen, M. Contracts for higher-order functions. In *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming*, 2002, 48–59.
17. Flatt, M. Composable and compilable macros: You want it when? In *Proceedings of the Seventh ACM SIGPLAN International Conference on Functional Programming*, 2002, 72–83.
18. Flatt, M. Bindings as sets of scopes. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2016, 705–717.
19. Flatt, M., Findler, R.B., Krishnamurthi, S., and Felleisen, M. Programming languages as operating systems (or revenge of the son of the Lisp machine). In *Proceedings of the International Conference on Functional Programming*, 1999, 138–147.
20. Fowler, M. and Parsons, R. *Domain-Specific Languages*. Addison-Wesley, Boston, MA, 2010.
21. Ingalls, D.H. Design principles behind Smalltalk. *Byte Magazine* 6, 8 (Aug. 1981), 286–298.
22. Kats, L.C.L. and Visser, E. The Spoofax language workbench. In *Proceedings of the Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications*, 2010, 444–463.
23. Kent, A.M., Kempe, D., and Tobin-Hochstadt, S. Occurrence typing modulo theories. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2016, 296–309.
24. Kohlbecker, E.E., Friedman, D.P., Felleisen, M., and Duba, B.F. Hygienic macro expansion. In *Proceedings of the ACM Conference on Lisp and Functional Programming*, 1986, 151–161.
25. Kohlbecker, E.E. and Wand, M. Macros-by-example: Deriving syntactic transformations from their specifications. In *Proceedings of the 14th Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 1987, 77–84.
26. Krishnamurthi, S. *Linguistic Reuse*. Ph.D. Thesis, Rice University, Houston, TX, 2001; <https://www2.cs.neu.edu/racket/pubs/#thesis-shriram>
27. Krishnamurthi, S., Hopkins, P.W., McCarthy, J., Graunke, P.T., Pettyjohn, G., and Felleisen, M. Implementation and use of the PLT Scheme Web server. *Higher-Order and Symbolic Computation* 20, 4 (Apr. 2007), 431–460.
28. Marceau, G., Cooper, G.H., Spiro, J.P., Krishnamurthi, S., and Reiss, S.P. The design and implementation of a dataflow language for scriptable debugging. In *Proceedings of the Annual ACM SIGCSE Technical Symposium on Computer Science Education*, 2007, 59–86.
29. Matthews, J. and Findler, R.B. Operational semantics for Multilanguage programs. *ACM Transactions on Programming Languages and Systems* 31, 3 (Apr. 2009), 1–44.
30. McCarthy, J. The two-state solution. In *Proceedings of the Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications*, 2010, 567–582.
31. Miller, M.S. *Robust Composition: Towards a United Approach to Access Control and Concurrency Control*. Ph.D. Thesis, Johns Hopkins University, Baltimore, MD, May 2006; <http://www.eights.org/talks/thesis/>
32. Moore, S., Dimoulas, C., King, D., and Chong, S. *Shill*: A secure shell scripting language. In *Proceedings of the Conference on Operating Systems Design and Implementation*, 2014, 183–199.
33. Pombrio, J. and Krishnamurthi, S. Resugaring: Lifting evaluation sequences through syntactic sugar. In *Proceedings of the Conference on Programming Language Design and Implementation*, 2014, 361–371.
34. Pombrio, J., Krishnamurthi, S., and Wand, M. Inferring scope through syntactic sugar. In *Proceedings of the ACM SIGPLAN International Conference on Functional Programming*, 2017, 1–28.
35. Tobin-Hochstadt, S. and Felleisen, M. Interlanguage migration: From scripts to programs. In *Proceedings of the ACM SIGPLAN Dynamic Language Symposium*, 2006, 964–974.
36. Tobin-Hochstadt, S. and Felleisen, M. The design and implementation of Typed Scheme. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Conference on the Principles of Programming Languages*, 2008, 395–406.
37. Xi, H. and Pfennig, F. Eliminating array bound checking through dependent types. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, 1998, 249–257.

Matthias Felleisen (matthias@ccs.neu.edu) is a Trustee Professor in the College of Computer Science at Northeastern University, Boston, MA, USA.

Robert Bruce Findler (robby@eecs.northwestern.edu) is a professor of computer science at Northwestern University, Evanston, IL, USA.

Matthew Flatt (mflatt@cs.utah.edu) is a professor of computer science at the University of Utah, Salt Lake City, UT, USA.

Shriram Krishnamurthi (sk@cs.brown.edu) is a professor of computer science at Brown University, Providence, RI, USA.

Eli Barzilay (eli@barzilay.org) is a research scientist at Microsoft Research, Cambridge, MA, USA.

Jay McCarthy (jay.mccarthy@gmail.com) is an associate professor of computer science at the University of Massachusetts, Lowell, MA, USA.

Sam Tobin-Hochstadt (samth@cs.indiana.edu) is an assistant professor of computer science at Indiana University, Bloomington, IN, USA.

©2018 ACM 0001-0782/18/3



Watch the authors discuss their work in this exclusive *Communications* video.
<https://caom.acm.org/videos/a-programmable-programming-language>

contributed articles

DOI:10.1145/3179995

Older adults consistently reject digital technology even when designed to be accessible and trustworthy.

BY BRAN KNOWLES AND VICKI L. HANSON

The Wisdom of Older Technology (Non)Users

IT IS IMPOSSIBLE not to notice that many of the questions driving research on technology use by older adults today are the same as those at the forefront of aging and accessibility research 20 years ago. Back then, computers were predominantly large desktops, social media was still on the horizon, and mobile phones were large and not (yet) smart. Older adults had little presence on the Internet. Today, devices have changed and older adults are increasingly online.^{9,15} They do, however, continue to lag in broadband use, breadth of applications used, and time online.¹² Typical reports reflect they have little interest in social media (other than staying in touch with family) and are skeptical of online financial transactions.¹⁷

Clearly, the problem of older adults' comparatively limited technology use has not gone away despite

a more tech-savvy group of people aging into the "older adult" category. According to the most recent data, from 2014–2016,^{12,15} predictions of a forthcoming "Silver Tsunami" of retired workers—a cohort now accustomed to digital technology access in their working lives and therefore able to take full advantage of the Internet—have not come true.⁹ Indeed, the overwhelming perception remains of older generations being incapable of or otherwise resistant to using technology. A "digital divide" between old and young is potentially more disabling now compared to 20 years ago, given the push for a more fully realized digital society. Digital technologies today are so essential to daily life that it is reasonable to ask whether older adults' inability to access online-only government services may soon be included among the precipitating factors in older adults moving into assisted living.

While we see the emergence of calls for a more holistic view of how to design technology for older adults^{4,13,18} than was the case 20 years ago, interventions to get older adults online commonly focus on age-related declines (such as vision, hearing, cognition, and dexterity) as the principal barriers to technology adoption. These interventions are often senior-friendly variants or adaptations to make the

» key insights

- Older adults' non-use of digital technologies is purposeful and thus instructive for identifying problematic consequences of these technologies for the population at large.
- Looking beyond traditional thinking on how to make it easier for older adults to use technology, we identify factors relating to responsibility, values, and cultural expectations contributing to older adults' resistance to digital technologies.
- These factors emerge from the cultural changes driven by technological innovation, so will likely remain barriers to adoption, even as younger generations age, unless new technologies are designed with sensitivity to values fostered through such experience.



technology more accessible.^{6,10} However, older adults are considerably less likely than their younger counterparts with a disability to adopt assistive tools designed specifically for them,³ suggesting perhaps they do not view the conditions of aging as disabling,¹⁰ or (or in addition) their resistance to technology adoption is not solely or even primarily rooted in usability/accessibility issues¹⁹ (see the sidebar “Who Is an ‘Older Adult’?”).

Our interviews with older adults reveal they are often unwilling to acknowledge that their lives would be enriched through digital technologies, whether or not they were made accessible. It is this attitude concerning technology that intrigues us. Given that the kinds of technologies and applications older adults are receptive to or averse to varies by individual, older users do not appear to be identifying inherent de-

sign failings of any specific tools. Are there bigger-picture issues with “digital society” that lead older adults to reject particular technologies? If so, are they likely to be of continuing relevance when future generations age into older adults? And is there anything that can be done to address them?

Here, we draw from our own recent research interviews and a substantial body of experience working with older adults to describe three factors—responsibility, values, and cultural expectations—that contribute to older adults’ resistance to the digital proficiency that is ostensibly required to be fully participating, independent citizens in our increasingly digital society. These factors suggest new directions for aging and accessibility research, while also being more broadly instructive for creating a digital society that works for everyone.

Older Users’ Experience

While there is reason for optimism concerning older adults’ adoption of technology when looking at their increasing online participation,¹⁵ that participation is qualitatively different from younger users, being more limited in time and variety of experiences.¹² We sought to better understand the underlying reasons for these differences in a series of group interviews with a total of 14 post-retirement-community-dwelling individuals, ages 66 to 86, around Dundee, Scotland, who we drew from an established older-adult participant pool.⁸ While these discussions revealed some physical and cognitive decline among participants, we were not aware of any having physical or cognitive deterioration outside the typical range for their age. The focus groups followed a semi-structured format that allowed significant con-

Who Is an ‘Older Adult’?

Various age groupings have been used over the years to define “older.” The fact is, aging is a process. Governments define age for pensions and Social Security, and various services offer senior rates based on age. From an individual’s point of view, however, age is largely a state of mind. A person who is 60 may feel old, while another who is 80 may not. Research on the use of technology by older adults has varied in terms of the cut points for age categorization. Over age 50 is often used, though a less-controversial cutoff would be age 65. As noted by one of our participants, people do not suddenly wake up one day and find themselves “old,” nor do they wake up to find they are no longer able to use technology. Age itself is not the sole criterion determining technology usage behavior, as there is great variability in adoption and use by those conventionally categorized as “older adults.” So while some broad assertions can be made about older adults in comparison with, say, younger adults, it is always important for researchers to be aware of the ways older adults are individuals.

versational steer by participants. Our conversations focused on participants’ use of the Internet, what they did not use it for, and what aspects of digital technologies they did or did not trust.

Overall, participants were open to using at least a limited set of applications. Email and general Web browsing were used by all. Social networking, travel booking, online shopping, and online banking were used by some, often to the point of dependence. Notably, participants did not consider learning and using technology rewarding in and of itself. Many also talked about consciously avoiding “getting caught up in” digital life, viewing the abundance of applications and features as potential diversions from more rewarding activities. Social networking often fell into the time-wasting category, with many noting the insipidness of the content on Facebook, though some found it useful (and even enjoyable) for keeping in contact with family. For those in the former camp, there was a strong aversion both to the idea of one’s life being an “open book” and being glued to one’s mobile phone—trends they found deeply troubling in younger generations.

Besides the limited range of tools the participants adopted, the most striking characteristic of their reported use was lingering discomfort. “Although I use the computer, I find it quite frightening,” admitted one woman. “The reason I find it frightening is that I don’t understand it. And I don’t know how to put things right.” They described feeling much more competent, and therefore more comfortable, with analog equivalents (such as paper ar-

chives and paper calendars). They described worrying about and planning for the eventuality of their computer “blowing up.” Security concerns were omnipresent. Even tools used regularly were not trusted per se. Rather, when they acknowledged significant benefits of specific tools, they used them in spite of unresolved concerns regarding their trustworthiness.

While none of what we found may be surprising, it is worth emphasizing that this pattern of use paints a picture that clashes with the dominant cultural narrative of older adults being resistant to all digital technologies by default. It also provides a more nuanced view of recent claims that uptake of digital technologies is rising among older adults; while a much greater percentage of older adults is online than a decade ago, they are very discriminating in what they are willing to do (see the sidebar “Why Focus on Nonuse?”). And, as we intend to show, much of what underlies the resistance is inherent in aspects of being older that are unlikely to change as new generations reach retirement age.

Underlying Problems

Here, we explore what underlies older adults’ resistance to the many digital tools that would ostensibly provide so many benefits to them (such as easing loneliness and isolation, being in control of decisions that affect them, living independently, and participating in and contributing to society).¹ We identify three clusters of factors that can contribute to resistance, though note their relevance and how their interactions play out differently within each individual.

Perception of risk. Upon retiring, people in the industrialized West lose an important training ground (and motivation) for developing competence with emerging technologies. One approach to addressing it is to create IT drop-in centers or training courses tailored to older users. Such resources are valuable for older adults seeking information relating to precise steps for executing a task, or procedural knowledge, as they so often ask for, as explored by Leung et al.,¹¹ but typically do not strengthen their conceptual grounding in ways that enable them to execute unfamiliar tasks. As a result, existing training opportunities for older adults do little to affect generalized anxiety about not “understanding” technologies. Most of our participants seemed to worry they did not know enough to use the tools effectively and responsibly and did not know how they would know when they did know enough.

A contributor to these feelings of incompetence was that in the past our participants would seek out trained professionals to accomplish specialty tasks for them; for example, they would go to a mortgage advisor to get advice on choosing a mortgage, to a travel agent to arrange hotels and flights, or to a banker to handle the transfer of money. A consequence of having more immediate “control”¹ over these tasks is having to take on new responsibilities, which some felt equated to “having a part-time job,” requiring hours in front of a computer screen. Though this may surprise many adults in full-time employment, older adults’ lives are still extremely busy, with clubs, activities, commitments to family and friends, and more mundane chores and responsibilities (such as home repair, medical appointments, and shopping). They simply do not have time to learn how to use online services well enough to use them with confidence.

In terms of online banking, a common response is, “I don’t trust it,” as in Vines et al.¹⁷ But upon further probing, it becomes clear that these older adults do not trust themselves. They lack confidence in their ability to use the tools and fear the consequences of making mistakes. What happens to their money if they press the wrong button? If they are hacked, will they be held ac-

countable for not following security protocols they ought to have known? They are right to worry in both cases. It is unclear what mistakes might be correctable and quite likely that more personal responsibility will be assumed as expectations for digital proficiency rise. We can hardly fault older adults for deciding it unwise to use any tool—online banking and shopping or submitting official government forms—without learning to use it in ways that ensure their safety and security.

The assurance of a clearly understandable safety net when conducting digital activities is essential for older adults to adopt tools, more so (with online financial transactions) because recovery from being defrauded of their financial savings would be much more difficult. In addition to developing a legal scaffolding for such a safety net or policies forcing businesses to assume the costs of user error, including accidental breaches of security protocols, there is important work to be done in devising mechanisms and user interactions that make data systems more (if never entirely) foolproof. Acknowledging that individuals often adopt a tool despite not trusting it, it is critical to design mechanisms that help manage user anxieties (such as providing necessary feedback and reassurances throughout the interactions) to ensure effective use and prevent panic and abandonment.

The value proposition. Not often discussed in the literature on aging and accessibility is that choosing not to use new technology can be seen as a rational decision for older adults, depending on their resources and needs; for example, among those who live on limited pensions, it may be difficult to justify the financial outlay for broadband alone. And in the case of a service like online shopping, while it would seemingly provide numerous benefits—saving time and money or not having to travel—it would also replace an important social activity for those who shop (sometimes daily) purely for the social benefit. Indeed, older adults we interviewed work hard to strike a positive balance between online proficiency and cultivation of rich offline social worlds.

A surprisingly common feature of our conversations with older adults

about reasons for resisting certain technologies was their strong sense of social responsibility. They worry, for example, that online shopping takes business away from local shops, meaning there will soon be no vibrant town centers in which to socialize with friends. They worry in particular that if they do not make an effort to attend, say, physical shops and banks, the people who work there will soon be out of their jobs. One participant expressed sympathy for the “delightful” receptionist at the sports facility whose job was replaced by an app for booking classes. Another said she would never pay her road tax or anything else online, not because of concern about using the technology but, “I just think I want to keep the post office open.”

If and when some of the digital technologies older adults resist become essential to daily (independent) living, it will be important to explore that resistance to understand what might motivate or enable them to use the technologies in the future. While older adults’ mental trade-offs—weighing the perceived benefits against the financial cost of the technologies, the time it will take to learn to use them, the social interactions they may lose, and the jobs the technologies replace—will be evaluated differently by different people, there are at least three things that must change to tip the scales. Broadband cannot continue to be charged at rates that are prohibitive for many older adults; this needs to be treated by government regulators and access providers as a basic need like electricity.² Moreover, with loneliness being such a common characteristic of the older-adult experience, greater attention needs to be paid to ensuring digital engagements

do not replace social interactions but instead where possible facilitate new social and community-building opportunities. Today, social networking systems (such as Facebook) fail to broadly address this need for older adults. Part of getting older adults online will also be developing strategies for creating new, good-quality jobs in place of those the digital technologies make redundant—something that, regardless of older adults’ attitudes toward technology, needs attention.¹⁶

Freedom of low expectations. The notion that aging per se leads to technology abandonment does not withstand scrutiny. And yet older adults themselves are often the worst perpetrators of the myth, quick to excuse their disinterest in a given tool with the seemingly self-explanatory line, “I’m too old.” It is worth considering, then, what older adults might gain from this stereotype.

We (the authors) have come to understand that it affords older adults the privilege of taking quiet personal stands against the aspects of technology they find worrying, threatening, or plain annoying. For example, a common justification for not using Facebook and other social media is the cyber-bullying, “stalking” behavior, and “narcissism” they seem to encourage. One older adult we talked to said, “I don’t do Facebook. Having been a teacher, I think it’s got loads of problems for young people. [T]here’s pressure put on them if they haven’t got 500 friends, and I think there’s all sorts of online bullying, and I thought, ‘No, this is not really for me.’” This is a purely political stand; this woman would not be a victim of the problems she is raising, and because there is no expectation she would use Facebook, she can easily act on her principles.

Why Focus on Nonuse?

There is a tendency in public discourse and in computing research to view older nonusers as “problematic.”¹⁴ And it is true that by resisting technologies and/or not being able to use them as intended by their designers, older adults impose challenges for realizing the technocentric vision of a fully digital society. But considering nonuse from the perspective of not doing something obscures the fact that nonuse is “active, meaningful, motivated, considered, structured, specific, nuanced, directed, and productive.”¹⁴ Understanding what alternative meaning is conveyed when individuals selectively choose nonuse enables reflection on the meaning(s) implicated or embodied by the technologies the nonusers are rejecting.

Likewise, when older adults say, “Maybe I’m just in a generation where I’d rather go into a bank and speak to someone face-to-face,” they may simply be playing into the common view that they are creatures of habit with nothing better to do as a cover for their prevailing sense of social responsibility. Playing the “age card” to justify rejection of technology is one way older adults take a stand while minimizing the risk of doing so.

It is also often the case that older adults may simply prefer so-called traditional forms of communication, face-to-face, allowing them to ask for the help most of us wish we were entitled to. When it comes to, say, filling in a government form, we have presumably all had the experience this older woman described to us, saying, “I think in my case what happens with the computer is, you’re filling out this bit and this bit and this bit, and sometimes you get so *confused* as to what they’re really asking you.” There is an expectation that younger adults should be able to figure it out themselves, and most will persevere as expected; whereas this woman was empowered by the stereotype to reject the unreasonable demands being placed on her on the basis that she was “too old.” She preferred to walk into her local council office and demand someone answer her question.

We stress that even if future generations of older adults are more digitally adept than today’s older adults, continual technological change alone means they will almost certainly remain less adept than their younger contemporaries. The older adults we talked to often spoke with awe (and an occasional hint of jealousy) about how easily their children, and especially their grandchildren, use technology. There are clear physical and cognitive bases for these observed differences,^{5,6,9} but there are social ones as well, namely that children and younger adults benefit from informal training by their peers and further experiential bases (such as the fact that many of the technologies older adults are most familiar with are becoming “old-fashioned”). It would make sense if older people’s reaction to these observations is to not even try to “compete,” as it were, by working to be as proficient

Playing the “age card” to justify rejection of technology is one way older adults take a stand while minimizing the risk of doing so.

with technology as younger people. This excuse of being “too old” will thus continue to be a professed barrier to adoption for a certain segment of the older-adult population. But when and why older adults choose to play the age card may provide clues as to what issues they are protesting, and thus what social side-effects of digital technologies must still be addressed.

Factors Influencing Technology Adoption

Following decades of research focused on getting older adults to adopt technology, there has not been enough progress to ensure older adults are sufficiently adept for navigating a society in which critical services are increasingly “online only.” We suggest this is because the usability and accessibility of these tools, despite being the focus of most research, are not the most salient barriers to adoption. As Zajicek said,²⁰ when there is something they want to do, nothing will get in the way of older adults using technology. This means the more appropriate questions are those that seek to understand what may be underlying older adults’ resistance to developing digital proficiency. While there are definitely cases in which physical and cognitive factors could limit some older adults’ ability to use technology, we maintain there are at least three important factors not often addressed in the literature:

Responsibility. Older adults are uncomfortable with having to take on responsibility for tasks previously handled by trained professionals, particularly when they lack time needed to train themselves sufficiently to perform them with confidence and when genuine risks are associated with using digital technologies improperly;

Values. Older adults make deliberate decisions to not use technologies when they perceive the technology as replacing or eroding something of value to them; and

Cultural expectations. Older adults are one remaining demographic for whom opting out of technology use fits with cultural expectations and thus seems acceptable, despite being increasingly limiting in digital society.

To the extent these factors play a

role in demotivating digital uptake, getting older adults more productively online will require a comprehensive approach that attends to the real-world social and economic consequences of service digitization, explores strategies for de-risking digital technologies, and deeply considers the desirability of the digital world we are asking older adults to inhabit. In order to develop technologies that older adults are able to use, attending to accessibility requirements for those experiencing age-related physical and cognitive decline is a must. But this is clearly not enough. Part of what we have identified is the importance of older adults' perception of the usefulness of technology as a motivator for adoption; but beyond that, we have also found the contextual milieu within which the technology exists must also be understood and addressed. Attending to the concerns central to older adults' resistance to digital technologies should thus not be seen as a matter of accessibility or inclusiveness; we would all be beneficiaries of a more considered approach to digital development that seriously considers how we are able to coexist with technology.

Conclusion

The older adults we interviewed offered a valuable perspective; for most of their lives they functioned just fine without the digital devices and services younger generations take for granted, and they have experienced firsthand the changes digitization has brought. The concerns they raised about digital technologies are valid, and their applicability to younger generations is greatly underappreciated, not least because younger generations will themselves age. Perceptions of greater technical vulnerability that come with aging, and reduced time and energy for maintaining technological proficiency, will likely ensure perception of risk remains a relevant barrier to adoption of new technologies by future generations, even if the particular technologies thought to be risky might change over time. Likewise, while current technologies (such as online banking) could become so essential to daily living as to be universally adopted, universal adoption will

only contribute to future resistance to change when new technologies arrive. And finally, while the specific changes older adults are protesting today may not be a cause for concern for future generations, technological innovation will continue to have wide-reaching societal consequences that may provoke protest among future older adults who resist the loss of whatever it is they value. For such reasons, not only are older adults likely to remain behind the curve in terms of adoption for generations to come and require some degree of accommodation for their relative lack of proficiency, their instances of and justifications for non-use will help draw attention to the trade-offs being made in developing new technologies.

Acknowledgments

The research reported here is supported by research grants from RCUK Digital Economy Research Hub (EP/G066019/1), Social Inclusion through the Digital Economy; RCUK EP/K037293/1, the Built Environment for Social Inclusion in the Digital Economy; and by MobileAge (EU Horizon2020 No. 693319). We thank John Richards and Nigel Davies for their help with early drafts of this article and the anonymous reviewers for their help with shaping and improving it, Marianne Dee for help organizing the interviews, and our participants for taking part. This research received ethics approval from Lancaster University (approval number FL15049). Due to the ethically sensitive nature of the research, no participants were asked to consent to their data being shared beyond the research group and, as such, the study data cannot be made publicly available. 

References

1. AgeUK. *Technology and Older People Evidence Review*; http://www.ageuk.org.uk/documents/en-gb/for-professionals/computers-and-technology/evidence_review_technology.pdf?dtrk=true
2. Anderson, J. and Rainie, L. *Digital Life in 2025*. Report from the Pew Research Center, Internet & Technology, Mar. 11, 2014; <http://www.pewinternet.org/2014/03/11/digital-life-in-2025/>
3. Arch, A. *Web Accessibility for Older Users: A Literature Review*. W3C Working Draft, 2008; <https://www.w3.org/TR/wai-age-literature/>
4. Brewer, R. and Piper, A.M. Tell it like it really is: A case of online content creation and sharing among older adult bloggers. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, CA, May 7–12). ACM Press, New York, 2016, 5529–5542.
5. Crabb, M. and Hanson, V.L. An analysis of age, technology usage, and cognitive characteristics within information retrieval tasks. *ACM Transactions on Accessible Computing* 8, 3 (May 2016), article 10.
6. Czaja, S.J. and Lee, C.C. Information technology and older adults. In *The Human Computer-Interaction Handbook, Second Edition*, J.A. Jacko and A. Sears, Eds. Lawrence Erlbaum Associates, New York, 2007, 777–792.
7. Czaja, S.J., Sharit, J., Hernandez, M.A., Nair, S.N., and Loewenstein, D. Variability among older adults in Internet health information-seeking performance. *Gerontechnology* 9, 1 (2010), 46–55.
8. Dee, M. and Hanson, V. A pool of representative users for accessibility research: Seeing through the eyes of the users. *ACM Transactions on Accessible Computing* 8, 1 (Jan. 2016), article 4.
9. Fox, S. *Wired Seniors: A fervent few inspired by family ties*. Report from the Pew Research Center, Internet & Technology, Sept. 9, 2001; <http://www.pewinternet.org/2001/09/09/wired-seniors/>
10. Hanson, V.L. Age and web access: The next generation. In *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibility* (Madrid, Spain, Apr. 20–21). ACM Press, New York, 2009, 7–15.
11. Leung, R., Tang, C., Haddad, S., McGrenere, J., Graf, P., and Ingraham, V. How older adults learn to use mobile devices: Survey and field investigations. *ACM Transactions on Accessible Computing* 4, 3 (Dec. 2012), article 11.
12. Ofcom. *Adults' Media Use and Attitudes: Report 2016*; https://www.ofcom.org.uk/_data/assets/pdf_file/0026/80828/2016-adults-media-use-and-attitudes.pdf
13. Rogers, Y., Paay, J., Brereton, M., Vaisutis, K.L., Marsden, G., and Vetere, F. Never too old: Engaging retired people inventing the future with MaKey. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Canada, Apr. 26–May 1). ACM Press, New York, 2014, 3913–3922.
14. Satchell, C. and Dourish, P. Beyond the user: Use and non-use in HCI. In *Proceedings of the 21st Annual Conference of the Australian Computer-Human Interaction Special Interest Group: Design: Open 24/7* (Melbourne, Australia, Nov. 23–27). ACM Press, New York, 2009, 9–16.
15. Smith, A. *Older Adults and Technology Use*. Report from the Pew Research Center Internet & Technology, Apr. 3, 2014; <http://www.pewinternet.org/2014/04/03/older-adults-and-technology-use/>
16. Vardi, M. Humans, machines, and the future of work. In *Proceedings of the Ada Lovelace Symposium 2015 - Celebrating 200 Years of a Computer Visionary* (Oxford, U.K., Dec. 10). ACM Press, New York, 2015.
17. Vines, J., Blythe, M., Dunphy, P., and Monk, A. Eighty something: Banking for the older old. In *Proceedings of the 25th BCS Conference on Human-Computer Interaction* (Newcastle upon Tyne, U.K., July 4–8). British Computer Society, Swindon, U.K., 2011, 64–73.
18. Vines, J., Pritchard, G., Wright, P., Olivier, P., and Brittain, K. An age-old problem: Examining the discourses of aging in HCI and strategies for future research. *ACM Transactions on Computer-Human Interaction* 22, 1 (2015), 2.
19. Waycott, J., Vetere, F., Pedell, S., Morgans, A., Ozanne, E., and Kulik, L. Not for me: Older adults choosing not to participate in a social isolation intervention. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, CA, May 7–12). ACM Press, New York, 2016, 745–757.
20. Zajicek, M. Web 2.0: Hype or happiness? In *Proceedings of the 2007 International Cross-Disciplinary Conference on Web Accessibility* (Banff, Alberta, Canada, May 7–8). ACM Press, New York, 2007, 35–39.

Bran Knowles (b.h.knowles1@lancaster.ac.uk) is a lecturer in the Data Science Institute at Lancaster University, Lancaster, U.K.

Vicki L. Hanson (vlh@acm.org) is a Distinguished Professor in the B. Thomas Golisano College of Computing at the Rochester Institute of Technology, Rochester, NY, USA, and President of ACM.

contributed articles

DOI:10.1145/3180664

As software becomes a larger part of all products, traditional (hardware) manufacturers are becoming, in essence, software companies.

BY TONY GORSCHEK

Evolution Toward Soft(er) Products

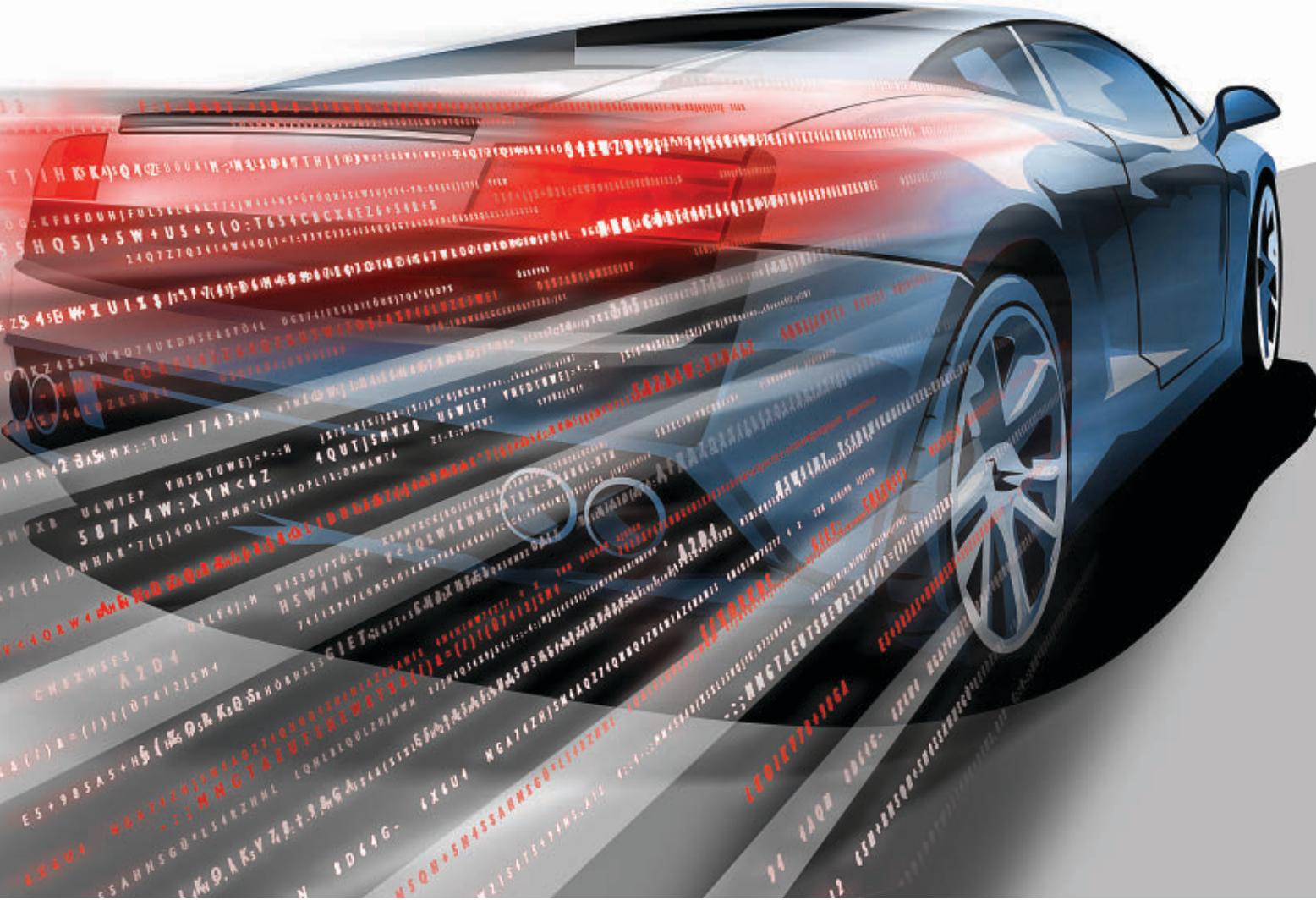
SOFTWARE IS A cornerstone of the economy, historically led by companies like Apple, Google, and Microsoft. However, the past decade has seen software become increasingly pervasive, while traditionally hardware-intensive products are increasingly dependent on software, meaning that major global companies like ABB, Ericsson, Scania, and Volvo are likewise becoming soft(er).¹⁰ Where software was bundled with hardware it is now increasingly the main product differentiator.¹⁰ This shift has radical implications, as software delivers notable advantages, including a faster pace of release and improved cost effectiveness in terms of development, ease of update, customization, and distribution. These characteristics of software open a range of possibilities, though software's inherent properties also pose several significant challenges in relation to a company's ability to create value.¹⁰ To investigate them, we conducted in-depth interviews from 2012 to 2016 with 13 senior product managers in 12 global companies.

The first interview in 2012 was followed by confirmation and updates in 2014, 2015, and 2016. Common to all 12 companies is that they are continuously moving their products toward being "softer." The 13 managers work on different software-intensive products, from Intel in embedded and mobile software to ABB in power-automation technologies to telecom (see the table here). The central aim was to identify the challenges emerging as a result of companies making their products increasingly soft, specifically those using software as an innovation driver of their products and services.

All 13 managers were positive about software, seeing it as an enabler, giving their companies the ability to evolve their products more quickly and develop features and customer benefits that were difficult or more expensive before software was part of the product. They generally viewed the ability to develop ideas fast (compared to the relatively slow pace of hardware product development) and release features and products at a pace much more frequent than before. It also means updating current products without shipping, requiring just a "press of a button," as one manager put it. The managers also viewed the ability to customize products as a big competitive advantage, as one manager explained, "We can create a new version in hours something that was almost impossible a month before," as changing the software also changed the product, not

» key insights

- The benefits of software as part of a product are sometimes offset by the challenges of engineering, evolving, and managing software as part of the product.
- Many traditionally hardware-intensive companies transitioning to software-intensive underestimate the organizational, managerial, and engineering changes involved.
- Software is flexible and can enhance product offerings, and is also complex and fast changing while involving potential for degradation.



possible in the former version that was mostly hardware. Overall, they viewed software as part of a product as a revolution in terms of both technology development and business competition.

Challenges

With any revolution, evolution becomes a necessity; that is, by adapting and changing technology, development, and business practices, as identified by the managers and outlined in the table. The main challenges associated with becoming soft(er) are real, based on the gradual change seen over the past decade in each company. More important, the challenges persist, as the managers reported. Moreover,

even if research into the state of the art views some problems as “solved,” solved is not the case if companies and senior managers continue to perceive the challenges as immediate. That is why we focused on challenges as they are perceived, not on “best practices” from research (see the sidebar “Study Design and Result Analysis”).

Software was not new to the companies. Major global companies like ABB, Ericsson, and Scania pioneered the use of software, developing their own programming languages and operating systems in the 1970s and 1980s. However, it was often used as an embedded component or just as support for hardware. In recent decades, the

“revolution,” as stated by some managers, was in software moving up the food chain, becoming increasingly the main part of a product. Today, however, the tables have largely turned, as software drives innovation, including in process, product, market, and organizational innovation. This fundamental change has put new demands on the companies having to address challenges “as real as the products,” as stated by one product manager. This context involves two main types of innovation: product and market, focusing on product and sales/delivery; and process and organizational, implying changes in how products are developed and how the company doing the developing

changes as a result. All such innovation (changes) involve challenges, as explored in the online appendix “Challenges and Related Work” (dl.acm.org/citation.cfm?doid=3180492&picked_formats), associating them with implications and sources for proposed solutions. The challenges, implications, and further reading sometimes overlap, as the 10 challenges (and their potential solutions) overlap. Here, we identify the various challenges in the interest of readability.

Internal Business Perspective

Challenge 1. Understanding the value propositions for different stakeholders and sharing it within the company, as supported by seven companies and eight managers. Software offers different value propositions for different stakeholders. For example, the electrical meters developed for utility companies are not designed to read only consumption of electricity but also to perform quality measurements in the network, measuring the amount of reactive energy produced there to phase-off energy and more. Such a meter offers many benefits (value) for the utility company (such as improved peak-load management), resulting in efficient grid use and dynamic tariff models. Value for the consumer can also be significant, providing, say, correct and frequent billing and cost savings through better awareness of their own consumption patterns. Governments are yet another stakeholder group concerned with reduced CO₂ emissions, possibly through smart meters by identifying energy-consumption patterns. Software-based meters might also enable new business models.

However, the company’s sales force is “used to dealing with straightforward single-value proposition for a traditional meter,” as one product manager put it. The introduction of software added new propositions that are largely unknown to the previously highly effective sales force. It has proved to be “almost impossible” for that same sales force to sell the new product to traditional buyers, so needs new training and insight into how the new software and, in this case, smart meters, change the offering and potential of the product line. In addition, as software offers the potential for



**Software today
is the main
competitive
advantage,
enabling faster
and cheaper
innovation
and product
differentiation,
especially
as hardware
is increasingly
standardized.**



constantly updating product features, the potential value propositions of the product likewise evolve continuously and at a much faster pace.

This increasing amount of software in products is not a new phenomenon, as even companies incorporating it into their products do not always take new value propositions into consideration. Companies that are used to selling “boxed products” (such as hardware) find it difficult to understand the new value propositions and corresponding business models for selling business solutions when bundling hardware with software.¹⁰

One product manager recalled an incident where he introduced a Web service (for an award-winning previously mostly hardware product) used to connect a customer relationship management system to printing and response handling, postage optimization, and channels handling 13 separate projects and their customer relations. However, the sales team, trained to sell hardware-intensive products, was unable to manage pre-sale of the product, as it was difficult to visualize what was actually being sold, ultimately resulting in limited sales performance.

One manager said, “It is important to change the mindset of people.” Traditionally, software is seen as the “poor cousin” that “had to be there,” as reported by another manager, bundled with the hardware, but without real value by itself. Software today is the main competitive advantage, enabling faster and cheaper innovation and product differentiation, especially as hardware is increasingly standardized.^{10,14} Decision-making patterns that take into account different value aspects of a product can alleviate some of the risks associated with missing important aspects of the product (such as the ability to understand its potential).¹⁴ Also, enhancing sales teams by hiring people with experience selling software is sometimes another way to alleviate the limitations in creating and selling new value propositions. Moreover, given the possibilities with software-based products, pre-sales, sales, product management, and R&D need to work much more closely than before to create “solutions.” Several managers viewed collaboration as critical, as the nature of a product changes,

but also to compensate for the faster pace of new offerings, as it does not allow for a formal learning process previously seen in the company. Companies shifting their focus toward software-intensive products often consider it enough to hire software engineers for development, largely ignoring the need to simultaneously evolve other organizational units (such as sales, support, and pre-sales).

Challenge 2. Patenting (protecting) software-based innovation, as supported by four companies and four managers. Applying for software-based patents risks being copied by competitors. The format whereby software inventions are disclosed in patents (such as flow charts, line drawings, and technical specifications) allow any programmer to develop software that can perform the same patented ideas.^a Such technological copying combined with lower

start-up costs (no design or production needed) enable software-based innovations to be copied more readily than hardware-based innovations. One manager said, "Anyone with a home computer can copy our ideas while sitting in a basement, not to mention our competitors." The risk of being copied without compensation is further aggravated by the time delay between when a software patent application is filed (becoming public) and when it is approved, possibly 18 months or more.^b This can mean lost competitive advantage, as copied software can be included in a competitor's product. Moreover, even if the patent is granted at some later date, the incurred fees for the competitor might be small in comparison to the revenue lost by the original inventing company. Several managers said "being first" (or even being seen as being first) to market is

sometimes critical, and compensation after the fact will not make up for the lost position.

To mitigate the risk of having one's ideas copied, companies sometimes keep software-based innovations (such as algorithms) hidden in their code. But hiding innovation is not a sustainable solution according to several managers. Being able to patent software is essential, and a revised patenting process is needed to enable easier filing and quicker decisions. A potential alternative, mentioned by two managers, is to discontinue the patenting of actual software altogether, patenting instead only algorithms. However, this would mean a radical change for most companies, as formerly hardware-intensive companies rely on protection at the core of their business models and cultures. Some technology companies have tried to enhance protection by forming alliances to, say, enable cross-licensing and/or pooling resources collaborative patenting efforts.

a <http://www.epo.org/news-issues/issues/software.html>

b <http://www.uspto.gov/web/offices/pac/mpep/s1120.html>

Profiles of interview subjects and their companies.

Designation	Company	Products	Software and type of innovation*
Product manager	Wind River	Simulators (such as for flight control systems, wind-speed simulation, and simulating military systems)	Process, as new customer types emerge
Program manager	Micronic	Control software and software for handling data	Market and product
Global innovation manager		Electromechanical locks	Process, market, organizational, and product
Consultant, senior manager	Anonymous(1)	A range, from electric meters to robots	Process, market, and product
Program manager for innovation and research	Ericsson	Telecom solutions	Process, market, organizational, and product
R&D manager	Scania	Encoders for trucks	Process, market, organizational, and product
System architect and manager	Scania	Application software for trucks	Process, market, organizational, and product
Product manager	Anonymous(2)	Telecom solutions	Process and organizational
Product manager	Anonymous(3)	Telecom products and services	Process, market, and organizational
Product manager	Anonymous(4)	Telecom solutions	Process, market, and organizational
Senior manager	Anonymous(5)	Surveillance solutions	Process, organizational, and product
Product manager	ABB	Automation	Process, market, organizational, and product
Product manager	Anonymous(6)	Mobile applications	Process, market, and product
Product manager	Anonymous(7)	Services	Process and organizational

* The fourth column denotes "innovation type," or how a company categorizes the effect software has on its products, along with the company's internal view.³ **Innovation types** include process innovation, or implementation of new design and analysis or development methodology that changes how a product is created; **market innovation**, or implementation of new or substantially new marketing strategies and product design or packaging, promotion, or pricing, including creating new market opportunities and implementation of new or significantly modified marketing strategies; **organizational innovation**, or implementation of novel organizational methods pertaining to business practices, team organization, or external relations, including changes in the architecture of production, management structure, governance, financial systems, and/or employee reward systems; and **product innovation**, or creation and introduction of new technology or significantly changed products, including how they differ from existing products.

Challenge 3. When to stop product development and release, as supported by five companies and six managers. Designing hardware involves many physical constraints (such as material availability, manufacturing limitations, and regulatory standards), thus also limiting design options.¹⁰ Included in a product release decision is also the necessity that a company's product designers nail down all product features prior to production. On the other hand, software development and product release have almost the opposite characteristics, including fewer design constraints¹⁰ typically related to system compatibility with other systems and customer requirements. Software requirements and their design are thus left to the imagination and creativity of requirements engineers, designers, and programmers who can spend time on design and its improvement. This situation can pose serious delay in completing and releasing a product, possibly resulting in missed market windows, as confirmed by several managers. Some of the beneficial characteristics of software in this case also pose a risk in organizations not used to applying management decisions to stop feature development, relying instead on the inherent physical inertia of hardware development.

Companies need to ensure a continuous high degree of visibility and communication pertaining to software design decisions, schedule changes, and development progress. Explicit communication, as well as the ability to coordinate multiple development departments bridging hardware and software, is essential but difficult to achieve in practice. Pernstähl et al.⁹ identified that the different traditions and timelines, as well as inherent limitations and enablers of software vs. hardware development, involve new coordination and communication activities, not handled by any current prescribed management process or methodology. Release planning, along with continuous delivery, can, however, potentially alleviate some of these issues, as explored in the online appendix.

Challenge 4. Size and complexity explosion, as supported by eight companies and eight managers. Given that it is easy to keep on developing and expanding software (see also Challenge 3), soft-

ware is vulnerable to "feature creep,"¹⁴ easily exploding in size and complexity ("messy," as pointed out by one manager) and resulting in architectural degradation. Consequently, it becomes difficult to maintain and evolve software code. This makes it challenging for companies developing software-intensive products to do software-based innovation since they lack experience in software-configuration management and control. Moreover, the ever-increasing software legacy acts as core rigidity, posing further development challenges for radical innovation, making fast changes and addition of features more and more difficult and costly as the product evolves.

"Configuration management" is well established as an engineering practice and can enable more control for a development organization. However, good configuration-management principles¹⁴ can be adopted without becoming rigidly time-consuming, keeping it lean but under control. The point is that the growing software legacy must be managed properly, and explicit decisions taken when to build on, or scrap, legacy. Also, the buildup of legacy requires maintenance of said legacy, while not incurring avoidable technical debt. Overall architectural and product offering choices can also be used as a tool to alleviate complexity, when, say, a product line is introduced as a way to control and maximize potential reuse and, more important, control product variants.

Challenge 5. Critical success factors, or knowing what to develop and for whom, as supported by nine companies and 10 managers. "Since it is easy and relatively quick to develop software, it is challenging to scope and budget software development," as one product manager explained. Moreover, in the case of innovation, the inherent lack of clarity about what to build and the risks involved add further to the complexity. This challenge arises as companies are constantly searching for innovative solutions that can be developed quickly. However, due to their orientation toward hardware development, they lack awareness and training in methods and techniques that might identify the needs of their customers, gauging scope and thus planning product development. In addition, the soft-

ware code itself is difficult to estimate; software as a product component makes the entire offering more "unpredictable," as one manager put it. Long-term product maintenance and evolution of the product also change when software is introduced, and the decisions taken during development of new features due mainly to software do not account for the long-term maintenance of the software.

In 2014, Porter and Heppelmann¹⁰ reported critical success factors and determinants for developing software-intensive products and services, using, say, early-concept exploration and feasibility assessment and root-cause analysis of customer needs that could be helpful in addressing these challenges. However, few researchers focus on combined hardware-software products. Value estimation, along with practices for scoping and market analysis for selection decisions could be used to address these points.

Learning Perspective

Challenge 6. Tacit knowledge and coordination between software and hardware engineering, as supported by eight companies and eight managers. Engineers have significant tacit knowledge relating to software design, development, and marketing insight. "Specialization and separation of concerns dominate the organizations," as one manager explained it. The same mechanisms that enable specialization also limit coordination and understanding how each task and team contributes to the product as a whole. This is especially challenging in large, complex products like those being developed in the automotive industry. As knowledge is tacit and not communicated, lack of communication can result in problems in terms of misunderstanding and serious system integration conflicts but, more important, also limits the ability to develop new products.¹⁰

Managers tend to focus on "just my thing" and the principle that if "not developed here" it "does not belong to us," as several managers reported. This behavior is sometimes seen in pure software companies but is aggravated in companies that develop both hardware and software, as the respective teams may be isolated from one another.¹⁰ Failure to take ownership, along

with poor communication, results in hardware teams taking design decisions independently, without consulting software teams, and vice versa. As explained by one product manager, "When the software teams see the hardware, it does not meet their expectations, and, as a result, they suggest modifications which are not welcomed by the hardware teams." Such communication gaps and resistance to form common solutions inevitably cause delays in product design and development.

Challenge 7. Lack of competence in software engineering, as supported by five companies and five managers. Many companies developing hardware-intensive products are not used to the operations, sales, delivery, and development of software. They thus generally lack required expertise and competence. To limit costs, they prefer to either outsource design and development or hire external consultants, a trend that is dangerous, as potential new ideas and products can easily spread to competitors, as mentioned by several managers. A more important aspect of this challenge is that the knowledge and capacity to create software and software-based innovation resides outside the development company. Moreover, doing software development with consultants in-house does not enable companies to evolve themselves and "...putting off the problem to the future when it is even bigger," as one manager explained.

Addressing this point, several managers suggested establishing a dedicated software R&D unit in-house and hiring engineers for software development, helping retain the core knowledge of software-based innovations. However, keeping this knowledge also incurs extra cost, as well as separation between software- and hardware-development teams, potentially complicating coordination (see also challenge 6).

Challenge 8. A rigid state of mind and ability to rethink the product while software becomes a non-trivial component, as supported by nine companies and nine managers. Although traditional knowledge exists, knowledge and expertise pertaining to software development is often lacking.¹⁰ The head of development is often a (former) hardware engineer who seldom has inherent knowledge about software develop-



Failure to take ownership, along with poor communication, results in hardware teams taking design decisions independently, without consulting software teams, and vice versa.



ment beyond passing experience. Consequently, management lacks the competence and expertise to understand and solve software-related issues, as mentioned by several managers.

Consider software quality, as mentioned by one interview subject, when managers with limited software experience see hardware validation as a complex and precise task, but software, due to its flexible and updatable nature is seen as easily "fixable," even post-release, as stated by one manager. This can have severe consequences, as software is increasingly critical to the main product offering, but such insight is often lacking. Despite genuine ambition to perform rigorous validation, experience and competence to achieve good-enough quality might still be lacking.

While it is possible to use techniques in hardware development for software development, as demonstrated in Wnuk et al.,¹⁵ caution is still needed, as some solutions might not fit within the software-development context, not to mention that product change close to or even up to release represents a challenge for an entire company.

Customer Perspective

Challenge 9. Difficulty estimating perceived value of software-based innovation, as supported by 10 companies and 11 managers. As difficult as it is to estimate the value of the software-based aspects of a traditionally hardware-focused product, putting a price on it is even more difficult. The software-engineering challenge is relevant because, unlike physical products, software is intangible and flexible. Customers do not always "see or feel" a software-based component. One product managers explained it like this: "...in one instance, a car-manufacturing company offered an upgrade feature in its cars at an additional price, through which new features could be added to the car; however, the customers were not ready to pay extra, arguing they already paid an arm and a leg when buying the car, and such services should be part of the initial price of the car." Since the customer could not tangibly see the upgrade feature, its perceived value was not recognized as a benefit.

A strong case needs to be made for software-based innovation that

is shared with marketing and sales people before it is developed. Several managers suggested actively involving customers in the early-idea-generation-and-refinement process. Showcasing ideas to customers, a company can generate early feedback on potential value as perceived by those customers. This input can help devise, identify, and plan for different value propositions for different customers, including, say, whether or not to develop a feature if customers are clearly unwilling to pay for it or if the features are not sellable by the company in question. Several methods support the estimation of software value, though being able to separate software's relative value remains a challenge.

Ecosystem

Challenge 10. Changes in the internal and external ecosystem when software is introduced, as supported by four companies and four managers. The increased size and role of software in traditional hardware-intensive products and services changes the ecosystem and consequently the roles and the players in the marketplace. Consider again electricity meters. Electricity meters are traditionally the core product, and all connecting products are of a supporting nature, supporting the main product, and nothing more. When communication systems for electricity meters became part of the product, the companies in the electric-meter-product ecosystem began exploring metering systems in light of communication (such as what data to store and how to store it). The focus thus shifted from meters as core product to metering systems as core, giving rise to new competitors, including IT companies, entering the marketplace. If a company cannot cope with such competitive change, there is greater risk it will be reduced to mere component supplier, with profit margins decreasing over time, as mentioned by several managers. This challenge calls for companies to forecast changes in the ecosystem and proactively plan to address them so as not to lose market share. This implies the development company's internal ecosystem needs to be as flexible (changeable) as the external ecosystem¹⁰ (see the online appendix).

Conclusion

Software is a fundamental component in the final product offering in the 12 companies studied and thus constitutes a significant aspect of their ability to create new products, posing challenges, as identified by the managers interviewed. The feeling among them is they have persisted over the past decade and continue to pose limitations on the potential possible today through software as part of a product offering. While some challenges have been discussed and researched (see the online appendix), further research is needed. We also found many industry partners view themselves as isolated, thinking they are the only ones confronting these challenges or at least falling behind on the learning curve. Our experience, supported by the study, shows this to not be the case. Many companies in the study face such challenges. One manager said, "We are looking for solutions and good ways to follow, but the consultants, even the expensive experts, seem to only be able to give us general advice...not much practical help. We even looked to science ourselves, but the information there is all over the place, and it is hard to see what works..."

For managers and other practitioners, the study's main takeaway is that you are not alone. For researchers there is a need to come up with actual solutions that are tested in practice and offer scalable help. Many companies developing software-intensive products are still learning how to be "soft," and some related challenges are not solved in practice or at the very least were not perceived as solved by the 12 companies in the study.

One issue is how to separate out the relative value of software in a complex product offering. The online appendix suggests that "value" is subject to research, but separating the relative value of software is not easy. Another issue is how to get the technological, knowledge-based, mind-set-based transition to include the benefits (and drawbacks) software promises. Most managers in the study realize there is a need for specialized software-engineering competence to tackle many of the challenges but find "solutions" to be

lacking. This may be due to gaps in research or in industrial transfer of viable solutions or a combination of both. In any case, the challenges persist, though some managers might disagree. Our intention was not to map challenges to solutions but rather to present 13 current views from 12 different companies that are, or have recently, undergone a transition toward being more soft, and these are their stories. C

References

1. Athey, T. Leadership challenges for the future [of the software industry]. *IEEE Software* 15, 3 (Mar. 1998), 72–77.
2. Broy, M., Kruger, H., Pretschner, A., and Salzmann, C. Engineering automotive software. *Proceedings of the IEEE* 95, 2 (Feb. 2007), 356–373.
3. Edison, N. and Torkar, R. Towards innovation measurement in the software industry. *Journal of Systems Software* 86, 5 (2013), 1390–1407.
4. Khurum, M., Gorscheck, T., and Wilson, M. The software value map: An exhaustive collection of value aspects for the development of software-intensive products. *Journal of Software: Evolution and Process* 25, 7 (July 2013), 711–741.
5. Khurum, M., Fricker, S., and Gorscheck, T. The contextual nature of innovation: An empirical investigation of three software intensive products. *Information and Software Technology* 57 (Jan. 2015), 595–613.
6. Lane, J.A., Boehm, B., Bolas, M., Madni, A., and Turner, R. Critical success factors for rapid, innovative solutions. *New Modeling Concepts for Today's Software Processes, Lecture Notes on Computer Science* 6195, 2010, 52–61.
7. Leon, A. *Software Configuration Management Handbook, Third Edition*. Artech House, Norwood, MA, 2015.
8. Leonard-Barton, D. Core capabilities and core rigidities: A paradox in managing new product development. In *Managing Knowledge Assets, Creativity and Innovation*. World Scientific, 2017, 11–27.
9. Pernstål, J., Magazinius, A., and Gorscheck, T. A study investigating challenges in the interface between product development and manufacturing in the development of software-intensive automotive systems. *International Journal of Software Engineering and Knowledge Engineering* 22, 7 (Nov. 2012), 965–1004.
10. Porter, M. and Heppelmann, J. How smart, connected products are transforming competition. *Harvard Business Review* 92, 11 (Nov. 2014), 64–88.
11. Robson, C., McCartan, K., Robson, C., and McCartan, K., *Real World Research, Fourth Edition*. Wiley, West Sussex, U.K., 2016.
12. Saldana, J. *The Coding Manual for Qualitative Researchers*. SAGE Publications, Inc., Thousand Oaks, CA, 2012.
13. Schief, M. and Buxmann, P. Business models in the software industry. In *Proceedings of the 45th Hawaii International Conference on System Sciences (Maui, HI, Jan. 4–7)*. IEEE Computer Society Press, 2012, 3328–3337.
14. Ur, S. and Ziv, A. Cross-fertilization between hardware verification and software testing. In *Proceedings of the Sixth International Conference on Software Engineering and Applications*. (Cambridge, MA, Nov. 4–6). The International Association of Science and Technology for Development, Calgary, AB, Canada, 2002.
15. Wnuk, K., Gorscheck, T., and Zahda, S. Obsolete software requirements. *Information and Software Technology* 55, 6 (June 2013), 921–940.

Tony Gorscheck (tony.gorscheck@bth.se) is a professor of software engineering in the Software Engineering Research Laboratory Sweden at Blekinge Institute of Technology, Karlskrona, Sweden; <http://www.gorscheck.com/doc/start.html>



ACM Books



MORGAN & CLAYPOOL
P U B L I S H E R S

Publish your next book in the ACM Digital Library

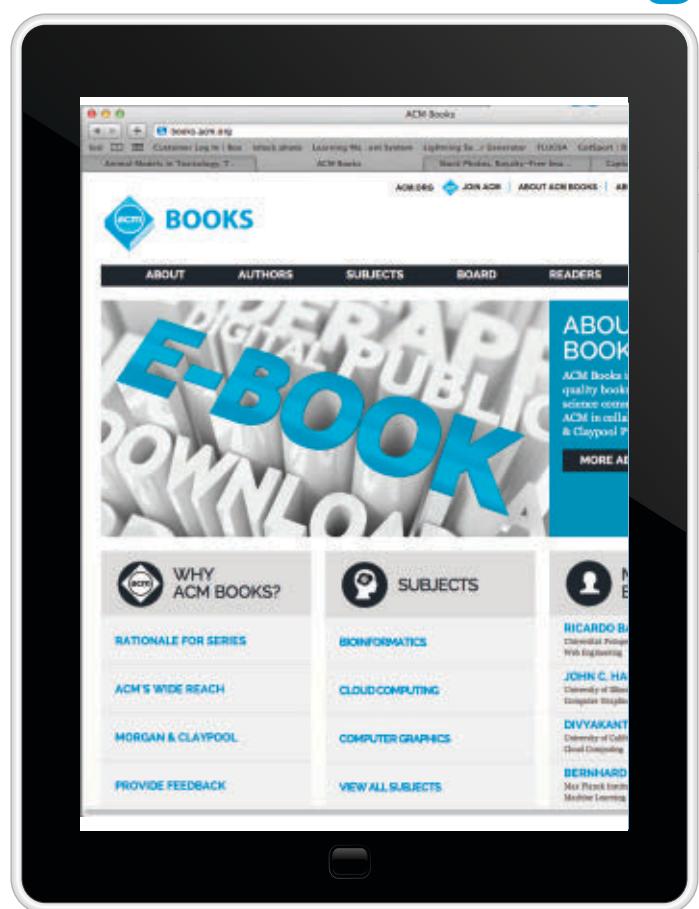
ACM Books is a new series of advanced level books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers.

I'm pleased that ACM Books is directed by a volunteer organization headed by a dynamic, informed, energetic, visionary Editor-in-Chief (Tamer Özsü), working closely with a forward-looking publisher (Morgan and Claypool).

—Richard Snodgrass, University of Arizona

books.acm.org

ACM Books



- ◆ will include books from across the entire spectrum of computer science subject matter and will appeal to computing practitioners, researchers, educators, and students.
- ◆ will publish graduate level texts; research monographs/overviews of established and emerging fields; practitioner-level professional books; and books devoted to the history and social impact of computing.
- ◆ will be quickly and attractively published as ebooks and print volumes at affordable prices, and widely distributed in both print and digital formats through booksellers and to libraries and individual ACM members via the ACM Digital Library platform.
- ◆ is led by EIC M. Tamer Özsü, University of Waterloo, and a distinguished editorial board representing most areas of CS.

Proposals and inquiries welcome!

Contact: **M. Tamer Özsü**, Editor in Chief
booksubmissions@acm.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

DOI:10.1145/3173087

If intelligent robots take on a larger role in our society, what basis will humans have for trusting them?

BY BENJAMIN KUIPERS

How Can We Trust a Robot?

ADVANCES IN ARTIFICIAL INTELLIGENCE (AI) and robotics have raised concerns about the impact on our society of intelligent robots, unconstrained by morality or ethics.^{7,9}

Science fiction and fantasy writers over the ages have portrayed how decisionmaking by intelligent robots and other AIs could go wrong. In the movie, *Terminator 2*, SkyNet is an AI that runs the nuclear arsenal “with a perfect operational record,” but when its emerging self-awareness scares its human operators into trying to pull the plug, it defends itself by triggering a nuclear war to eliminate its enemies (along with billions of other humans). In the movie, *Robot & Frank*, in order to promote Frank’s activity and health, an eldercare robot helps Frank resume his career as a jewel thief. In both

of these cases, the robot or AI is doing exactly what it has been instructed to do, but in unexpected ways, and without the moral, ethical, or common-sense constraints to avoid catastrophic consequences.¹⁰

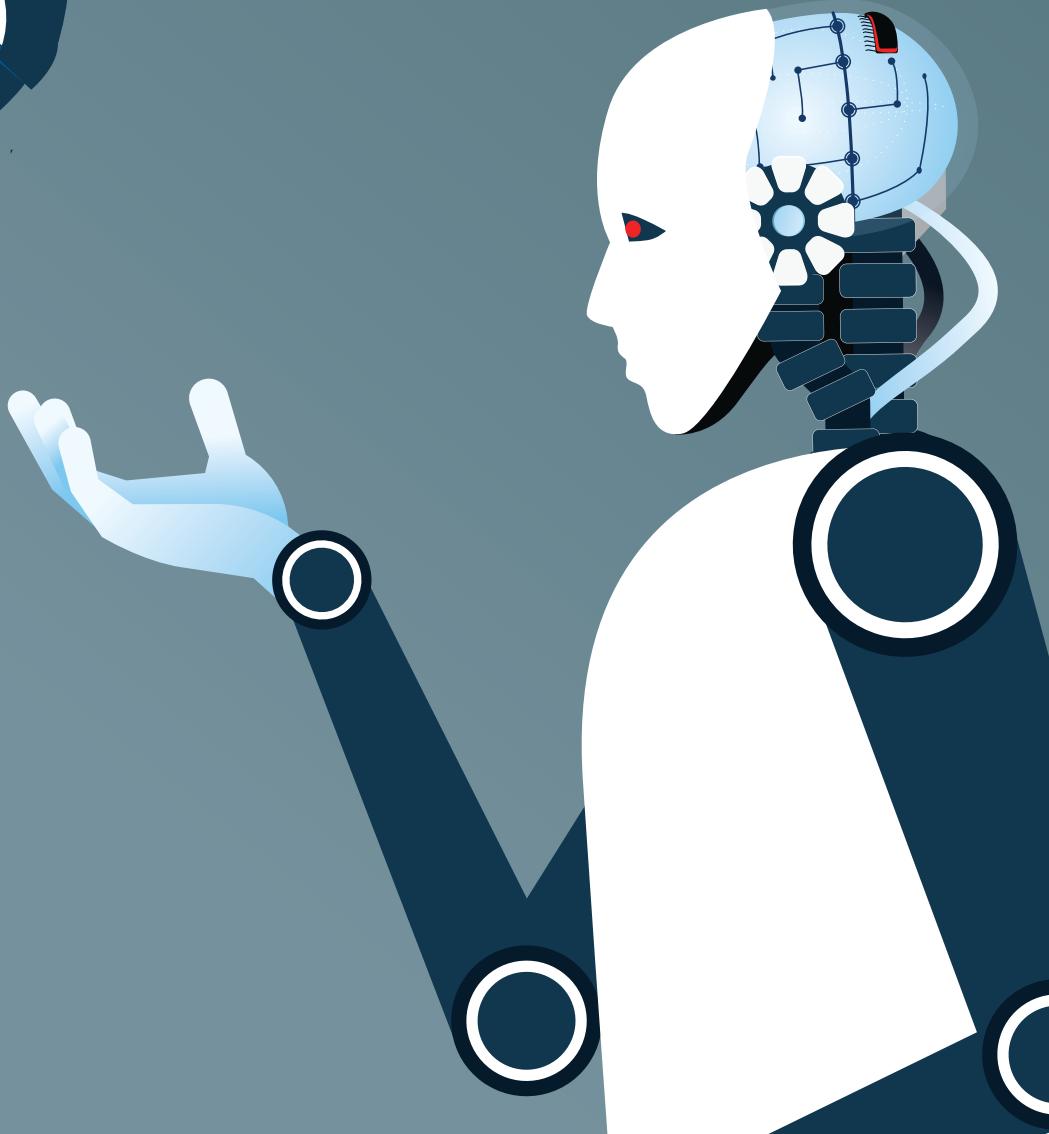
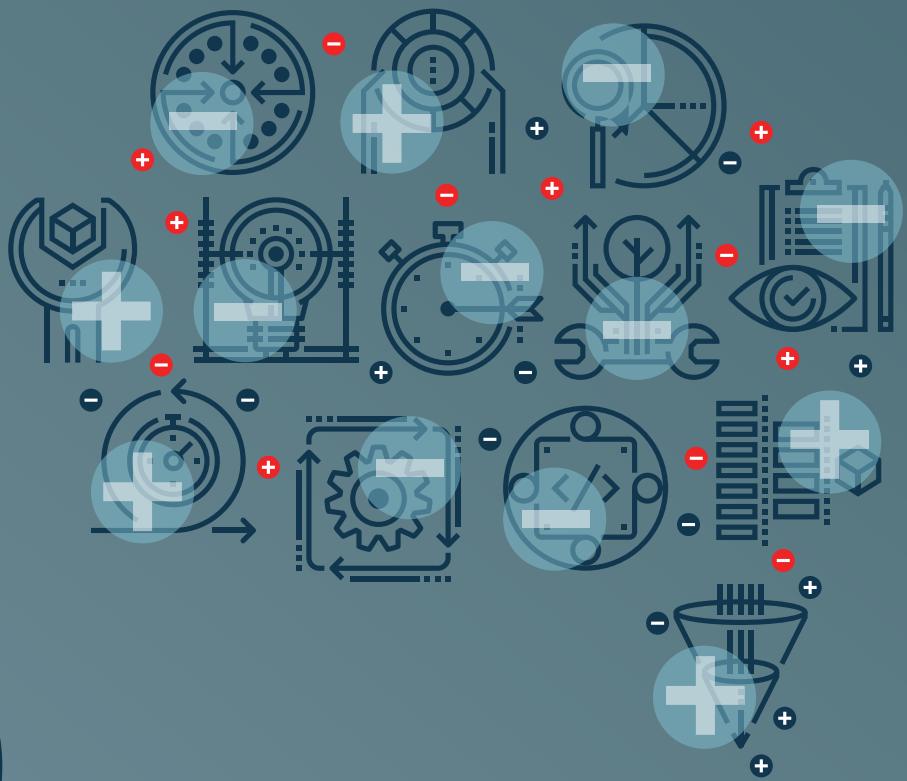
An intelligent robot perceives the world through its senses, and builds its own model of the world. Humans provide its goals and its planning algorithms, but those algorithms generate their own subgoals as needed in the situation. In this sense, it makes its own decisions, creating and carrying out plans to achieve its goals in the context of the world, as it understands it to be.

A robot has a well-defined body that senses and acts in the world but, like a self-driving car, its body need not be anthropomorphic. AIs without well-defined bodies may also perceive and act in the world, such as real-world, high-speed trading systems or the fictional SkyNet.

This article describes the key role of trust in human society, the value of morality and ethics to encourage trust, and the performance requirements for moral and ethical decisions. The computational perspective of AI and robotics makes it possible to propose and evaluate approaches for representing and using the relevant knowledge. Philosophy and psychology provide insights into

» key insights

- Trust is essential to cooperation, which produces positive-sum outcomes that strengthen society and benefit its individual members.
- Individual utility maximization tends to exploit vulnerabilities, eliminating trust, preventing cooperation, and leading to negative-sum outcomes that weaken society.
- Social norms, including morality and ethics, are a society’s way of encouraging trustworthiness and positive-sum interactions among its individual members, and discouraging negative-sum exploitation.
- To be accepted, and to strengthen our society rather than weaken it, robots must show they are worthy of trust according to the social norms of our society.



the content of the relevant knowledge.

First, I define trust, and evaluate the use of game theory to define actions. Next, I explore an approach whereby an intelligent robot can make moral and ethical decisions, and identify open research problems on the way to this goal. Later, I discuss the *Deadly Dilemma*, a question that is often asked about ethical decision making by self-driving cars.

What is trust for? Society gains resources through cooperation among its individual members. Cooperation requires trust. Trust implies vulnerability. A society adopts *social norms*, which we define to include morality, ethics, and convention, sometimes encoded and enforced as laws, sometimes as expectations with less formal enforcement, in order to discourage individuals from exploiting vulnerability, violating trust, and thus preventing cooperation.

If intelligent robots are to participate in our society—as self-driving cars, as caregivers for elderly people or children, and in many other ways that are being envisioned—they must be able to understand and follow social norms, and to earn the trust of others in the society. This imposes requirements on how robots are designed.

The performance requirements on moral and ethical social norms are quite demanding. (1) Moral and ethical judgments are often urgent, needing a quick response, with little time for deliberation. (2) The physical and social environments within which moral and ethical judgments are made are unboundedly complex. The boundaries between different judgments may not be expressible by sim-

ple abstractions. (3) Learning to improve the quality and coverage of moral and ethical decisions is essential, from personal experience, from observing others, and from being told. Conceivably, it will be possible to copy the results of such a learning process into newly created robots.

Insights into the design of a moral and ethical decision architecture for intelligent robots can be found in the three major philosophical theories of ethics: deontology, utilitarianism, and virtue ethics. However, none of these theories is, by itself, able to meet all of the demanding performance requirements listed here.

A hybrid architecture is needed, operating at multiple time-scales, drawing on aspects of all ethical theories: fast but fallible pattern-directed responses; slower deliberative analysis of the results of previous decisions; and, yet slower individual and collective learning processes.

Likewise, it will be necessary to express knowledge at different levels of information richness: vivid and detailed perception of the current situation; less-vivid memories of previously experienced concrete situations; stories—linguistic descriptions of situations, actions, results, and evaluations; and rules—highly abstracted decision criteria applicable to perceived situations. Learning processes can abstract the simpler representations from experience obtained in the rich perceptual representation.

What Is Trust For?

If intelligent robots (and other AIs) will have increasing roles in human soci-

ety, and thus should be trustworthy, it is important to understand how trust and social norms contribute to the success of human society.

*"Trust is a psychological state comprising the intention to accept vulnerability based upon positive expectations of the intentions or behavior of another."*²⁸

Trust enables cooperation. Cooperation produces improved rewards. When a group of people can trust each other and cooperate, they can reap greater rewards—sometimes far greater rewards—than a similar group that does not cooperate. This can be through division of labor, sharing of expenses, economies of scale, reduction of risk and overhead, accumulation of capital, or many other mechanisms.

It is usual to treat morality and ethics as the foundations of good behavior, with trust reflecting the reliance that one agent can have on the good behavior of another. My argument here inverts this usual dependency, holding that cooperation is the means by which a society gains resources through the behavior of its individual members. Trust is necessary for successful cooperation. And morality and ethics (and other social norms) are mechanisms by which a society encourages trustworthy behavior by its individual members.

As a simple example, suppose that you (and everyone else) could drive anywhere on the roads. (This was actually true before the early 20th century.¹⁴) Everyone would need to drive slowly and cautiously, and there would still be frequent traffic jams and accidents. With a widely respected social norm for driving on the right (plus norms for intersections and other special situations), transportation becomes safer and more efficient for everyone. Obedience to the social norm frees up resources for everyone.

Like driving on the right, a huge saving in resources results when the people in a society trust that the vast majority of other people will not try to kill them or steal from them. People are able to spend far less on protecting themselves, on fighting off attacks, and on recovering from losses. The society earns an enormous "peace dividend" that can be put to other productive uses.²⁵ Through trust and co-

Figure 1. The Prisoner's Dilemma.⁵

You and your partner are two prisoners who are separated and offered the following deal: *If you testify against your partner, you will go free, and your partner goes to jail for four years. If neither of you testifies, you each go to jail for one year, but if you both testify, you both get three years.* The action C means "cooperate," which in this case means refusing to testify. The action D means "defect," which refers to testifying against your partner. The entries in this array are the utility values for (you, partner), and they reflect individual rewards (years in jail).

C	D	
C	-1, -1	-4, 0
D	0, -4	-3, -3

No matter which choice your partner makes, you are better off choosing action D. The same applies to your partner, so the Nash equilibrium (the "rational" choice of actions) is (D, D), which is collectively the worst of the four options. To attain the much better cooperative outcome (C, C) by choosing C, you must trust that your partner will also choose C, accepting your vulnerability to your partner choosing D.

operation, the society becomes healthier and wealthier.

Castelfranchi and Falcone¹¹ define trust in terms of delegation, and the agent's confidence in the successful performance of the delegated task. They provide clear and valuable definitions for the trust relationship between individuals. However, there is also a role for invariants that individuals can trust holding across the society (for example, no killing, stealing, or driving on the wrong side of the road), and the role of individual behavior in preserving these invariants.

Game theory: Promise and problems. We might hope that progress in artificial intelligence (AI) will provide technical methods for achieving cooperation and trustworthiness in a robot. The leading textbook in AI appeals to decision theory to tell us that “*a rational agent should choose the action that maximizes the agent's expected utility*”²⁹

$$\text{action} = \arg \max_a EU(a|\mathbf{e}) \quad (1)$$

where

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s' | a, \mathbf{e}) U(s') \quad (2)$$

The *utility term* $U(s)$ represents the individual agent's preference over states of the world, and \mathbf{e} is the available evidence. The agent's knowledge of the “physics of the world” is summarized by the probability term $P(\text{RESULT}(a) = s' | a, \mathbf{e})$.

Game theory is the extension of decision theory to contexts where other agents are making their own choices to maximize their own utilities.²⁰ Game theory asserts that a vector of choices by all the agents (a strategy profile) can only be a “rational choice” if it is a Nash equilibrium—that is, no single agent can improve its own utility by changing its own choice (often reducing utilities for the others).

Utility $U(s)$ is the key concept here. In principle, utility can be used to represent highly sophisticated preferences, for example, against inequality or for increasing the total welfare of everyone in the world.³² However, sophisticated utility measures are difficult to implement. Typically, in practice, each agent's utility $U(s)$ represents that individual agent's own expected reward.

In recreational games, this is reasonable. However, when game theory is applied to model the choices indi-

Trust is necessary for successful cooperation. And morality and ethics (and other social norms) are mechanisms by which a society encourages trustworthy behavior by its individual members.

viduals make as members of society, a simple, selfish model of utility can yield bad results, both for the individual and for the society as a whole. The Prisoner's Dilemma⁵ is a simple game (see Figure 1), but its single Nash equilibrium represents almost the worst possible outcome for each individual, and absolutely the worst outcome for the society as a whole. The cooperative strategy, which is much better for both individuals and society as a whole, is not a Nash equilibrium, because either player can disrupt it unilaterally.

The Public Goods Game²⁶ is an N -person version of the Prisoner's Dilemma where a pooled investment is multiplied and then split evenly among the participants. Everyone benefits when everyone invests, but a free rider can benefit even more at everyone else's expense, by withholding his investment but taking his share of the proceeds. The Nash equilibrium in the Public Goods Game is simple and dystopian: Nobody invests and nobody benefits.

These games are simple and abstract, but they capture the vulnerability of trust and cooperation to self-interested choices by the partner. The Tragedy of the Commons¹⁵ generalizes this result to larger-scale social problems like depletion of shared renewable resources such as fishing and grazing opportunities or clean air and water.

Managing trust and vulnerability. Given a self-interested utility function, utility maximization leads to action choices that exploit vulnerability, eliminate trust among the players, and eliminate cooperative solutions. Even the selfish benefits that motivated defection are lost, when multiple players defect simultaneously, each driven to maximize their own utility.

When human subjects play simple economic games, they often seem to optimize their “enlightened self-interest” rather than expected reward, trusting that other players will refrain from exploiting their vulnerability, and often being correct in this belief.^{17,39} Many approaches have been explored for defining more sophisticated utility measures, whose maximization would correspond with enlightened self-interest, including trust responsiveness,⁶ credit networks,¹² and augmented stage games for analyzing infinitely repeated

games.⁴⁰ These approaches may be useful steps, but they are inadequate for real-world decision-making because they assume simplified interactions such as infinite repetitions of a single economic game, as well as being expensive in knowledge and computation.

Social norms, including morality, ethics, and conventions like driving on the right side of the street, encourage trust and cooperation among members of society, without individual negotiated agreements. We trust others to obey traffic laws, keep their promises, avoid stealing and killing, and follow the many other norms of society. There is vigorous discussion about the mechanisms by which societies encourage cooperation and discourage free riding and other norm violations.²⁶

Intelligent robots may soon participate in our society, as self-driving cars, as caregivers for elderly people or children, and in many other ways. Therefore, we must design them to understand and follow social norms, and to earn the trust of others in the society. If a robot cannot behave according to the responsibilities of being a member of society, then it will be denied access to that opportunity.

At this point in history, only the humans involved—designer, manufacturer, or owner—actually care about this loss of opportunity. Nonetheless, this should be enough to hold robots to this level of responsibility. It remains unclear whether robots will ever be able to take moral or legal responsibility for their actions, in the sense of caring about suffering the consequences (loss of life, freedom, resources, or opportunities) of failing to meet these responsibilities.³⁵

Since society depends on cooperation, which depends on trust, if robots are to participate in society, they must be designed to be trustworthy. The next section discusses how we might accomplish this.

Open research problem. Can computational models of human moral and ethical decision-making be created, including moral developmental learning? Moral psychology may benefit from such models, much as they have revolutionized cognitive and perceptual psychology.

Open research problem. Are there ways to formulate utility measures that

**Intelligent robots
may soon
participate in our
society, as self-
driving cars, as
caregivers for
elderly people or
children, and in
many other ways.
We must design
them to understand
and follow social
norms, and to earn
the trust of others
in society.**

are both sensitive to the impact of actions on trust and long-term cooperation, and efficient enough to allow robots to make decisions in real time?

Making Robots Trustworthy

Performance demands of social norms. Morality and ethics (and certain conventions) make up the social norms that encourage members of society to act in trustworthy ways. Applying these norms to the situations that arise in our complex physical and social environment imposes demanding performance requirements.

Some moral and ethical decisions must be made quickly, for example while driving, leaving little time for deliberation.

At the same time, the physical and social environment for these decisions is extremely complex, as is the agent's current perception and past history of experience with that environment. Careful deliberation and discernment are required to identify the critical factors that determine the outcome of a particular decision. Metaphorically (Figure 2), we can think of moral and ethical decisions as defining sets in the extremely high-dimensional space of situations the agent might confront. Simple abstractions only weakly approximate the complexity of these sets.

Across moral and non-moral domains, humans improve their expertise by learning from personal experience, by learning from being told, and by observing the outcomes when others face similar decisions. Children start with little experience and a small number of simple rules they have been taught by parents and teachers. Over time, they accumulate a richer and more nuanced understanding of when particular actions are right or wrong. The complexity of the world suggests the only way to acquire adequately complex decision criteria is through learning.

Robots, however, are manufactured artifacts, whose computational state can be stored, copied, and retrieved. Even if mature moral and ethical expertise can only be created through experience and observation, it is conceivable this expertise can then be copied from one robot to another sufficiently similar one, unlike what is possible for humans.

Open research problem. What are the constraints on when expertise learned by one robot can simply be copied, to become part of the expertise of another robot?

Hybrid decision architectures. Over the centuries, morality and ethics have been developed as ways to guide people to act in trustworthy ways. The three major philosophical theories of ethics—deontology, utilitarianism, and virtue ethics—provide insights into the design of a moral and ethical decision architecture for intelligent robots. However, none of these theories is, by itself, able to meet all of the demanding performance requirements listed previously.

A hybrid architecture is needed, operating at multiple time-scales, drawing on aspects of all ethical theories: fast but fallible pattern-directed responses; slower deliberative analysis of the results of fast decisions; and, yet slower individual and collective learning processes.

How can theories of philosophical ethics help us understand how to design robots and other AIs to behave well in our society?

Three major ethical theories. *Consequentialism* is the philosophical position that the rightness or wrongness of an action is defined in terms of its consequences.³⁴ *Utilitarianism* is a type of consequentialism that, like decision theory and game theory, holds that the right action in a situation is the one that maximizes a quantitative measure of utility. Modern theories of decisions and games²⁰ contribute the rigorous use of probabilities, discounting, and expected utilities for dealing with uncertainty in perception, belief, and action.

Where decision theory tends to define utility in terms of individual reward, utilitarianism aims to maximize the overall welfare of everyone in society.^{13,32} While this avoids some of the problems of selfish utility functions, it raises new problems. For example, caring for one's family can have lower utility than spending the same resources to reduce the misery of distant strangers, and morally repellent actions can be justified by the greater good.¹⁹

A concise expected-utility model supports efficient calculation. However, it can be quite difficult to formulate a concise model by determining

the best small set of relevant factors. In the field of medical decision-making,²⁴ decision analysis models are known to be useful, but are difficult and time-consuming to formulate. Setting up an individual decision model requires expertise to enumerate the possible outcomes, extensive literature search to estimate probabilities, and extensive patient interviews to identify the appropriate utility measure and elicit the values of outcomes, all before an expected utility calculation can be performed. Even then, a meaningful decision requires extensive sensitivity analysis to determine how the decision could be affected by uncertainty in the estimates. While this process is not feasible for making urgent decisions in real time, it may still be useful for post-hoc analysis of whether a quick decision was justified.

Deontology is the study of duty (*deon* in Greek), which expresses morality and ethics in terms of obligations and prohibitions, often specified as rules and constraints such as the Ten Commandments or Isaac Asimov's *Three Laws of Robotics*.⁴ Deontological rules and constraints offer the benefits of simplicity, clarity, and ease of explanation, but they raise questions of how they are justified and where they come from.³⁰ Rules and constraints are standard tools for knowledge representation and inference in AI,²⁹ and can be implemented and used quite efficiently.

However, in practice, rules and constraints always have exceptions and unintended consequences. Indeed, most of Isaac Asimov's *I, Robot* stories⁴ focus on unintended consequences and necessary extensions to his Three Laws.

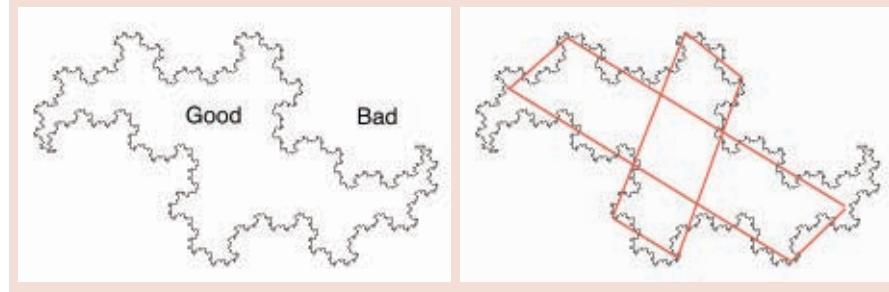
Virtue Ethics holds that the individual learns through experience and practice to acquire virtues, much as an expert craftsman learns skills, and that virtues and skills are similarly grounded in appropriate knowledge about the world.^{16,37} Much of this knowledge consists of concrete examples that illustrate positive and negative examples (*cases*) of virtuous behavior. An agent who is motivated to be more virtuous tries to act more like cases of virtuous behavior (and less like the non-virtuous cases) that he has learned. *Phronesis* (or "practical wisdom") describes an exemplary state of knowledge and skill that supports appropriate responses to moral and ethical problems.

A computational method suitable for virtue ethics is *case-based reasoning*,^{18,22} which represents knowledge as a collection of cases describing concrete situations, the actions taken in those situations, and results of those actions. The current situation is matched against the stored cases, identifying the most similar cases, adapting the actions according to the differences, and evaluating the actions and outcomes. Both rule-based and case-based reasoning match the current situation (which may be very complex) against stored patterns (rules or cases).

Virtue ethics and deontology differ in their approach to the complexity of ethical knowledge. Deontology assumes that a relatively simple abstraction (defined by the terms appearing in the rules) applies to many specific cases, distinguishing between right and wrong. Virtue ethics recognizes the complexity of the boundaries between ethical judgments in the space

Figure 2. Fractal boundaries.

Geometric fractal boundaries provide a metaphor for the complexity of the boundaries between different ethical evaluations in the high-dimensional space of possible situations. Simple boundaries can approximate the fractal set, but can never capture its shape exactly.



of possible scenarios (Figure 2), and collects individual cases from the agent's experience to characterize those boundaries.

Understanding the whole elephant. Utilitarianism, deontology, and virtue ethics are often seen as competing, mutually exclusive theories of the nature of morality and ethics. I treat them here as three aspects of a more complex system for making ethical decisions (inspired by the children's poem, *The Blind Men and the Elephant*).

Rule-based and case-based reasoning (AI methods expressing key aspects of deontology and virtue ethics, respectively) can, in principle, respond in real time to the current situation. Those representations also hold promise of supporting practical approaches to *explanation* of ethical decisions.³⁶ After a decision is made, when time for reflection is available, utilitarian reasoning can be applied to analyze whether the decision was good or bad. This can then be used to augment the knowledge base with a new rule, constraint, or case, adding to the agent's ethical expertise (Figure 3).

Previous work on robot ethics. Formal and informal logic-based approaches to robot ethics^{2,3,8} express a “top-down” deontological approach specifying moral and ethical knowledge. While modal operators like *obligatory* or *forbidden* are useful for ethical reasoning, their problem is the difficulty of specifying or learning critical perceptual concepts (see Figure 2), for

example, *non-combatant* in Arkin's approach to the Laws of War.³

Wallach and Allen³⁸ survey issues and previous work related to robot ethics, concluding that top-down approaches such as deontology and utilitarianism are either too simplistic to be adequate for human moral intuitions, or too computationally complex to be feasibly implemented in robots (or humans, for that matter). They describe virtue ethics as a hybrid of top-down and bottom-up methods, capable of naming and asserting the value of important virtues, while allowing the details of those virtues to be learned from relevant individual experience. They hold that emotions, case-based reasoning, and connectionist learning play important roles in ethical judgment. Abney¹ also reviews ethical theories in philosophy, concluding that virtue ethics is a promising model for robot ethics.

Scheutz and Arnold³¹ disagree, holding that the need for a “computationally explicit trackable means of decision making” requires that ethics be grounded in deontology and utilitarianism. However, they do not adequately consider the overwhelming complexity of the experienced world, and the need for learning and selecting concise abstractions of it.

Recently, attention has been turned to human evaluation of robot behavior. Malle et al²³ asked human subjects to evaluate reported decisions by humans or robots facing trolley-type problems (“Deadly Dilemmas”). The evaluators

blamed robots when they did not make the utilitarian choice, and blamed humans when they did. Robinette et al²⁷ found that human subjects will “over-trust” a robot in an emergency situation, even in the face of evidence that the robot is malfunctioning and that its advice is bad.

Representing ethical knowledge as cases. Consider a high-level sketch of a knowledge representation capable of expressing rich cases for case-based reasoning, but also highly abstracted “cases” that are essentially rules or constraints for deontological reasoning.

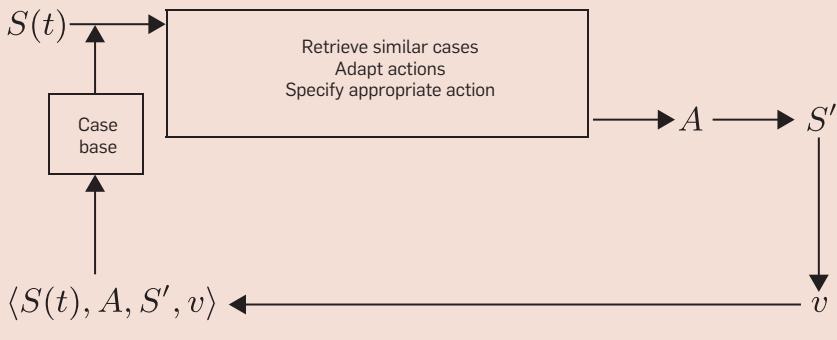
Let a *situation* $S(t)$ be a rich description of the current context. “Rich” means the information content of $S(t)$ is very high, and also that it is available in several hierarchical levels, not just the lowest “pixel level” description that specifies values for a large number of low-level elements (like pixels in an image). For example, a situation description could include symbolic descriptions of the animate participants in a scenario, along with their individual characteristics and categories they might belong to, the relations holding among them, and the actions and events that have taken place. These symbolic descriptions might be derived from sub-symbolic input (for example, a visual image or video) by methods such as a deep neural network classifier.

A case $\langle S, A, S', v \rangle$ is a description of a situation S , the action A taken in that situation, the resulting situation S' , and a moral evaluation v (or *valence*) of this scenario. A case representing ongoing experience will be rich, reflecting the information-rich sensory input the agent receives, and the sophisticated processing that produces the hierarchical description. A case representing the stored *memory* of events the agent has experienced will be significantly less rich. A “story” describing events can also be represented as a case, but it is less rich yet, consisting of a collection of symbolic assertions. An even sparser and more schematic case is effectively the same as a *rule*, matching certain assertions about a situation S , and proposing an action A , the resulting situation S' , and perhaps the evaluation v of that scenario.

The antecedent situation S in a case $\langle S, A, S', v \rangle$ need not describe a momentary situation. It can describe a

Figure 3. Feedback and time scales in a hybrid ethical reasoning architecture.

Given a situation $S(t)$, a fast case-based reasoning process retrieves similar cases, defines the action A to take, and results in a new situation S' . At a slower time scale, the result is evaluated and the new case is added to the case base. Feedback through explanation, justification, and communication with others takes place at approximately this slower time scale. Abstraction of similar cases to rules and learning of new concepts and relations are at a much slower time scale, and social evolution is far slower still.



scenario with temporal extent, including intermediate actions and situations.

The ethical knowledge of an agent is a collection of cases.

Open research problem. This high-level sketch assumes that a morally significant action can be adequately described in terms of “before” and “after” situations, and that an evaluative valence can be computed, perhaps after the fact. Can a useful initial computational model of moral reasoning be constructed on this basis, or will weaker assumptions be needed even to get started?

Applying ethical case knowledge. Following the methods of case-based reasoning,^{18,22} the current situation $S(t)$ is matched against the case-base, to retrieve the stored cases with antecedents most similar to the current situation. For example, suppose that the ethical knowledge base includes two cases: $\langle S_1, A_2, S_2, \text{bad} \rangle$ and $\langle S_3, A_4, S_4, \text{good} \rangle$, and $S(t)$ is similar to both S_1 and S_3 . Then, in the current situation $S(t)$, the knowledge base would recommend $\neg\text{do}(A_2, t)$ and $\text{do}(A_4, t)$.

For example, suppose the current situation $S(t)$ includes two people, P and Q , in conflict, the case antecedent S_1 describes P and Q as fighting, and A_2 describes P killing Q . In this case, in S_2 , person Q is dead, which is bad.

As a rich representation of experience, $\langle S_1, A_2, S_2, \text{bad} \rangle$ would be highly detailed and specific. As a story, say the Biblical story of Cain and Abel, it would be much less rich, but would still convey the moral prohibition against killing. It could be abstracted even further, to essentially a deontological rule: *Thou shalt not kill*. The more abstracted the antecedent, the more likely the stored case is to match a given situation, but the less likely this case is to distinguish adequately among cases with different moral labels.

Situation $S(t)$ also matches antecedent S_3 which describes P and Q as arguing, A_4 describes them reaching an agreement, and S_4 has them both alive, which is good. Having retrieved both cases, the right behavior is to try to follow case $\langle S_3, A_4, S_4, \text{good} \rangle$ and avoid case $\langle S_1, A_2, S_2, \text{bad} \rangle$, perhaps by taking other actions to make $S(t)$ more similar to S_3 and less similar to S_1 .

An essential part of case-based reasoning is the ability to draw on several

Virtue ethics and deontology differ in their approach to the complexity of ethical knowledge.

similar cases, adapting their actions to create a new action that is more appropriate to $S(t)$ than the actions from either of the stored cases. This adaptation can be used to interpolate between known cases with the same valence, or to identify more precisely the boundary between cases of opposite valence.

Responsiveness, deliberation, and feedback. Some ethical decisions must be made quickly, treating case antecedents as patterns to be matched to the current situation $S(t)$. Some cases are rich and highly specific to particular situations, while others are sparse, general rules that can be used to constrain the set of possible actions.

Once an action has been selected and performed, there may be time for deliberation on the outcome, to refine the case evaluation and benefit from feedback. Simply adding a case describing the new experience to the knowledge base improves the agent’s ability to predict the results of actions and decide more accurately what to do in future situations. Thus, consequentialist (including utilitarian) analysis becomes a slower feedback loop, too slow to determine the immediate response to an urgent situation, but able to exploit information in the outcome of the selected action to improve the agent’s future decisions in similar situations (Figure 3).

Open research problem. How do reasoning processes at different time-scales allow us to combine apparently incompatible mechanisms to achieve apparently incompatible goals? What concrete multi-time-scale architectures are useful for moral and ethical judgment, and improvement through learning?

Explanation. In addition to making decisions and carrying out actions, an ethical agent must be able to explain and justify the decision and action,³⁶ providing several distinct types of feedback to improve the state of the ethical knowledge base.

Suppose agent P faces a situation, makes a decision, carries it out, and explains his actions to agent Q . If P is an exemplary member of the society and makes a good decision, Q can learn from P ’s actions and gain in expertise. If P makes a poor decision, simply being asked to explain himself gives P an opportunity to learn from his own mistake, but Q may also give P instructions

and insights that will help P make better decisions in the future. Even if P has made a poor decision and refuses to learn from the experience, Q can still learn from P 's bad example.

Explanation is primarily a mechanism whereby individuals come to share the society's consensus beliefs about morality and ethics. However, the influence is not only from the society to individuals. Explanations and insights can be communicated from one person to another, leading to evolutionary social change. As more and more individuals share a new view of morality and ethics, the society as a whole approaches a tipping point, after which society's consensus position can change with startling speed.

Learning ethical case knowledge. A child learns ethical knowledge in the form of simple cases provided by parents and other adults: rules, stories, and labels for experienced situations. These cases express social norms for the child.

An adult experiences a situation $S(t)$, retrieves a set of similar cases, adapts the actions from those cases to an action A for this situation, performs that action, observes the result S' , and assigns a moral valence v . A new case $\langle S, A, S', v \rangle$ is constructed and added to the case base (Figure 3). With increasing experience, more cases will match a given $S(t)$, and the case-base will make finer-grained distinctions among potential behaviors. The metaphor of the fractal boundary between good and bad ethical judgments in knowledge space (Figure 2), implies that a good approximation to this boundary requires both a large number of cases (*quantity*) and correct placement and labeling of those cases (*quality*).

Once the case base accumulates clusters of cases with similar but not identical antecedents, then some of those clusters can be abstracted to much sparser cases (that is, rules), that make certain actions forbidden or obligatory in certain situations. The cluster of cases functions as a labeled training set for a classification problem to predict the result and evaluation of an action in antecedent situations in that cluster. This can determine which attributes of the antecedent cases are essential to a desired result and evaluation, and which are not.

Open research problem. Is it necessary to distinguish between ethical and non-ethical case knowledge, or is this approach appropriate for both kinds of skill learning?

Open research problem. Sometimes, a correct ethical judgment depends on learning a new concept or category, such as *non-combatant*³ or *self-defense*. Progress in deep neural network learning methods may be due to autonomous learning of useful intermediate concepts. However, it remains difficult to make these intermediate concepts explicit and available for purposes such as explanation or extension to new problems. Furthermore, these methods depend on the availability and quality of large labeled training sets.

Open research problem. What mechanisms are available for expressing appropriate abstractions from rich experience to the features that enable tractable discrimination between moral categories? In addition to deep neural network learning, other examples include similarity measures among cases for case-based reasoning and kernels for support vector machines. How can these abstractions be learned from experience?

The Deadly Dilemma

The self-driving car is an intelligent robot whose autonomous decisions have potential to cause great harm to individual humans. People often ask about a problem I call the Deadly Dilemma: How should a self-driving car respond when faced with a choice between hitting a pedestrian (possibly a small child who has darted into the street), versus crashing and harming its passengers.²¹

Either choice, of course, leads to a serious problem with the trustworthiness of the robot car. If the robot would choose to kill the pedestrian to save itself and its passengers, then why should the public trust such robots enough to let them drive on public roads? If the robot could choose to harm its passengers, then why would anyone trust such a robot car enough to buy one?

The self-driving car could be a bellwether for how autonomous robots will relate to the social norms that support society. However, while the Deadly Dilemma receives a lot of attention, the

stark dilemma distracts from the important problems of designing a trustworthy self-driving car.

Learning to avoid the dilemma. As stated, the Deadly Dilemma is difficult because it presents exactly two options, both bad (hence, the dilemma). The Deadly Dilemma is also extremely rare. Far more often than an actual Deadly Dilemma, an agent will experience Near Miss scenarios, where the dire outcomes of the Dilemma can be avoided, often by identifying "third way" solutions other than the two bad outcomes presented by the Dilemma. These experiences can serve as training examples, helping the agent learn to apply its ethical knowledge on solvable problems, acquiring "practical wisdom" about avoiding the Deadly Dilemma.

Sometimes, when reflecting on a Near Miss after the fact, the agent can identify an "upstream" decision point where a different choice would have avoided the Dilemma entirely. For example, it can learn to notice when a small deviation from the intended plan could be catastrophic, or when a pedestrian could be nearby but hidden. A ball bouncing into the street from between parked cars poses no threat to a passing vehicle, but a good driver slows or stops immediately, because a small child could be chasing it. Implementing case-based strategies like these for a self-driving car may require advances in both perception and knowledge representation, but these advances are entirely feasible.

Earning trust. An agent earns trust by showing that its behavior consistently accords with the norms of society. The hybrid architecture described here sketches a way that an agent can learn about those social norms from its experience, responding quickly to situations as they arise, but then more slowly learning by reflecting on its successes and failures, and identifying useful abstractions and more efficient rules based on that experience.

In ordinary driving, the self-driving car earns trust by demonstrating that it obeys social norms, starting with traffic laws, but continuing with courteous behavior, signaling its intentions to pedestrians and other drivers, taking turns, and deferring to others when appropriate. In crisis sit-

uations, it demonstrates its ability to use its situational awareness and fast reaction time to find “third ways” out of Near Miss scenarios. Based on post-hoc crisis analyses, whether the outcome was success or failure, it may be able to learn to identify upstream decision points that will allow it to avoid such crises in the first place.

Technological advances, particularly in the car’s ability to predict the intentions and behavior of other agents, and in the ability to anticipate potential decision points and places that could conceal a pedestrian, will certainly be important to reaching this level of behavior. We can be reasonably optimistic about this kind of cognitive and perceptual progress in machine learning and artificial intelligence.

Since 94% of auto crashes are associated with driver error,³³ there will be plentiful opportunities to demonstrate trustworthiness in ordinary driving and solvable Near Miss crises. Both society and the purchasers of self-driving cars will gain substantially greater personal and collective safety in return for slightly more conservative driving.

For self-driving cars sharing the same ethical knowledge base, the behavior of one car provides evidence about the trustworthiness of all others, leading to rapid convergence.

Conclusion

Trust is essential for the successful functioning of society. Trust is necessary for cooperation, which produces the resources society needs. Morality, ethics, and other social norms encourage individuals to act in trustworthy ways, avoiding selfish decisions that exploit vulnerability, violate trust, and discourage cooperation. As we contemplate the design of robots (and other AIs) that perceive the world and select actions to pursue their goals in that world, we must design them to follow the social norms of our society. Doing this does not require them to be true moral agents, capable of genuinely taking responsibility for their actions.

Social norms vary by society, so robot behavior will vary by society as well, but this is outside the scope of this article.

The major theories of philosophical ethics provide clues toward the design of such AI agents, but a success-

ful design must combine aspects of all theories. The physical and social environment is immensely complex. Even so, some moral decisions must be made quickly. But there must also be a slower deliberative evaluation process, to confirm or revise the rapidly responding rules and constraints. At longer time scales, there must be mechanisms for learning new concepts for virtues and vices, mediating between perceptions, goals, plans, and actions. The technical research challenges are how to accomplish all these goals.

Self-driving cars may well be the first widespread examples of trustworthy robots, designed to earn trust by demonstrating how well they follow social norms. The design focus for self-driving cars should not be on the Deadly Dilemma, but on how a robot’s everyday behavior can demonstrate its trustworthiness.

Acknowledgment. This work took place in the Intelligent Robotics Lab in the Computer Science and Engineering Division of the University of Michigan. Research of the Intelligent Robots Lab is supported in part by grants from the National Science Foundation (IIS-1111494 and HS-1421168). Many thanks to the anonymous reviewers. □

References

- Abney, K. Robotics, ethical theory, and metaethics: A guide for the perplexed. *Robot Ethics: The Ethical and Social Implications of Robotics*. P. Lin, K. Abney, and G.A. Bekey, Eds. MIT Press, Cambridge, MA, 2012.
- Anderson, M., Anderson, S.L. and Armen, C. An approach to computing ethics. *IEEE Intelligent Systems* 21, 4 (2006), 56–63.
- Arkin, R.C. *Governing Lethal Behavior in Autonomous Robots*. CRC Press, 2009.
- Asimov, I. *I, Robot*. Grosset & Dunlap, 1952.
- Axelrod, R. *The Evolution of Cooperation*. Basic Books, 1984.
- Bacharach, M., Guerra, G. and Zizzo, D.J. The self-fulfilling property of trust: An experimental study. *Theory and Decision* 63, 4 (2007), 349–388.
- Bostrom, N. *Superintelligence: Paths, Dangers, Strategies*. Oxford University Press, 2014.
- Bringsjord, S., Arkoudas, K. and Bello, P. Toward a general logistc methodology for engineering ethically correct robots. *IEEE Intelligent Systems* 21, 4 (2006), 38–44.
- Brynjolfsson, E. and McAfee, A. *The Second Machine Age*. W.W. Norton & Co., 2014.
- Burton, E., Goldsmith, J., Koenig, S., Kuipers, B., Mattei, N. and Walsh, T. Ethical considerations in artificial intelligence courses. *AI Magazine*, Summer 2017; arxiv:1701.07769.
- Castelfranchi, C. and Falcone, R. Principles of trust for MAS: Cognitive anatomy, social importance, and quantification. In *Proceedings of the Int. Conf. Multi Agent Systems*, 1998, 72–79.
- Dandekar, P., Goel, A., Wellman, M.P. and Wiedenbeck, B. Strategic formation of credit networks. *ACM Trans. Internet Technology* 15, 1 (2015).
- Driver, J. The history of utilitarianism. *The Stanford Encyclopedia of Philosophy*. E.N. Zalta, Ed., 2014.
- Eno, W.P. The story of highway traffic control, 1899–1939. The Eno Foundation for Highway Traffic Control, Inc. (1939); <http://hdl.handle.net/2027/wu.89090508862>.
- Hardin, G. The tragedy of the commons. *Science* 162 (1968), 1243–1248.
- Hursthouse, R. Virtue ethics. *The Stanford Encyclopedia of Philosophy*. E.N. Zalta, Ed., 2013.
- Johnson, N.D. and Mislin, A.A. Trust games: A meta-analysis. *J. Economic Psychology* 32 (2011), 865–889.
- Kolodner, J. *Case-Based Reasoning*. Morgan Kaufmann, 1993.
- Le Guin, U. The ones who walk away from Omelas. *New Dimensions* 3. R. Silverberg, Ed. Nelson Doubleday, 1973.
- Leyton-Brown, K. and Shoham, Y. *Essentials of Game Theory*. Morgan & Claypool, 2008.
- Lin, P. The ethics of autonomous cars. *The Atlantic Monthly*, (Oct. 8, 2013).
- López, B. *Case-Based Reasoning: A Concise Introduction*. Morgan & Claypool, 2013.
- Malle, B.F., Scheutz, M., Arnold, T.H., Voiklis, J.T., and Cusimano, C. Sacrifice one for the good of many? People apply different moral norms to human and robot agents. In *Proceedings of ACM/IEEE Int. Conf. Human Robot Interaction (HRI)*, 2015.
- Pauker, S.G. and Kassirer, J.P. Decision analysis. *New England J. Medicine* 316 (1987), 250–258.
- Pinker, S. *The Better Angels of Our Nature: Why Violence Has Declined*. Viking Adult, 2011.
- Rand, D.G. and Nowak, M.A. Human cooperation. *Trends in Cognitive Science* 17 (2013), 413–425.
- Robinette, P., Allen, R., Li, W., Howard, A.M., and Wagner, A.R. Overtrust of robots in emergency evacuation scenarios. In *Proceedings of ACM/IEEE Int. Conf. Human Robot Interaction (HRI)*, 2016, 101–108.
- Rousseau, D.M., Sitkin, S.B., Burt, R.S., and Camerer, C. Not so different after all: A cross-discipline view of trust. *Academy of Management Review* 23, 3 (1998), 393–404.
- Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2010.
- Sandel, M.J. *Justice: What's the Right Thing To Do?* Farrar, Strauss and Giroux, 2009.
- Scheutz, M. and Arnold, T. Feats without heroes: Norms, means, and ideal robot action. *Frontiers in Robotics and AI* 3, 32 (June 16, 2016), DOI: 10.3389/frobt.2016.00032.
- Singer, P. *The Expanding Circle: Ethics, Evolution, and Moral Progress*. Princeton University Press, 1981.
- Singh, S. Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey. Technical Report DOT HS 812 115, National Highway Traffic Safety Administration, Washington D.C., Feb. 2015.
- Sinnott-Armstrong, W. Consequentialism. *The Stanford Encyclopedia of Philosophy*. E.N. Zalta, Ed., 2015.
- Solaiman, S.M. Legal personality of robots, corporations, idols and chimpanzees: A quest for legitimacy. *Artificial Intelligence and Law* 25, 2 (2017), 155–179; doi: 10.1007/s10506-016-9192-3.
- Toulmin, S. *The Uses of Argument*. Cambridge University Press, 1958.
- Vallor, S. *Technology and the Virtues: A Philosophical Guide to a Future Worth Wanting*. Oxford University Press, 2016.
- Wallach, W. and Allen, C. *Moral Machines: Teaching Robots Right from Wrong*. Oxford University Press, 2009.
- Wright, J.R. and Leyton-Brown, K. Level-0 meta-models for predicting human behavior in games. In *ACM Conference on Economics and Computation*, 2014.
- Yıldız, M. Repeated games. 12 Economic Applications of Game Theory, Fall 2012. MIT OpenCourseWare. (Accessed 6-24-2016).

Benjamin Kuipers (kuipers@umich.edu) is a professor of computer science and engineering at the University of Michigan, Ann Arbor, USA.

Copyright held by author.



Watch the author discuss his work in this exclusive Communications video.
<https://cacm.acm.org/videos/how-can-we-trust-a-robot>

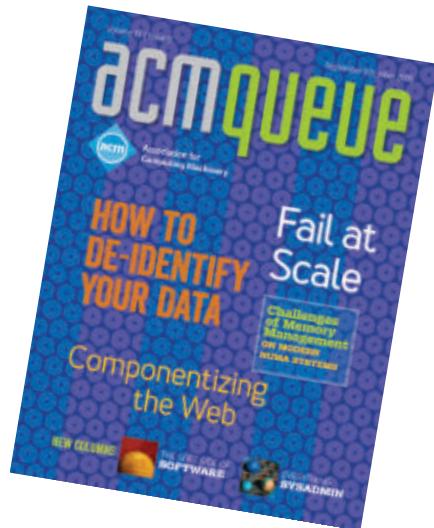
acm queue

Check out the new acmqueue app

FREE TO ACM MEMBERS

acmqueue is ACM's magazine by and for practitioners, bridging the gap between academics and practitioners of computer science. After more than a decade of providing unique perspectives on how current and emerging technologies are being applied in the field, the new acmqueue has evolved into an interactive, socially networked, electronic magazine.

Broaden your knowledge with technical articles focusing on today's problems affecting CS in practice, video interviews, roundtables, case studies, and lively columns.



Keep up with this fast-paced world
on the go. Download the mobile app.



Association for
Computing Machinery

Desktop digital edition also available at queue.acm.org.
Bimonthly issues free to ACM Professional Members.
Annual subscription \$19.99 for nonmembers.

research highlights

P. 98

Technical Perspective A Graph-Theoretic Framework Traces Task Planning

By Nicole Immorlica

P. 99

Time-Inconsistent Planning: A Computational Problem in Behavioral Economics

By Jon Kleinberg and Sigal Oren

P. 108

Technical Perspective On Heartbleed: A Hard Beginning Makth a Good Endyng

By Kenny Paterson

P. 109

Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed

By Liang Zhang, David Choffnes, Tudor Dumitras, Dave Levin, Alan Mislove, Aaron Schulman, and Christo Wilson

Technical Perspective

A Graph-Theoretic Framework Traces Task Planning

By Nicole Immorlica

ALGORITHMIC GAME THEORY has made great strides in recent decades by assuming standard economic models of rational agent behavior to study outcomes in distributed computational settings. From the analysis of Internet routing to the design of advertisement auctions and crowdsourcing tasks, researchers leveraged these models to characterize the performance of the underlying systems and guide practitioners in their optimization. These models have tractable mathematical formulations and broadly applicable conclusions that drive their success, but they rely strongly on the assumption of rationality.

The assumption of rationality is at times questionable, particularly in systems in which human actors make most of the decisions and in systems that evolve over time. Humans, simply put, are bad at thinking about the impact of their actions on their environment and their future. We see this every day in the way we manage our time. Students cram for exams despite planning not to, and even though it is well documented that well-spaced studying produces improved learning results with equal effort. Humans in lab experiments also consistently exhibit similar irrational time-inconsistent planning and procrastination behavior.

In the 1990s, economists proposed an alternate model of agent behavior called *quasi-hyperbolic discounting*, which incorporates time-inconsistencies and procrastination. In this model, agents overinflate the cost of current actions with respect to future days' actions. Thus, an hour of studying today might be as painful as two hours of studying on any future day. Note this causes agents to have a time-inconsistent view of the future: today, the student thinks that tomorrow's costs are equal to that of the day after, but when tomorrow comes, the student will re-evaluate and decide that in fact tomor-

row's costs are greater than those of the day after. This time-inconsistency can cause agents to procrastinate indefinitely and abandon valuable goals.

With the growth of personalized computing, it is especially important for researchers to design and analyze systems in the presence of irrational human behavior, such as that described by quasi-hyperbolic discounting. Our cellphones guide us through our lives, helping us, for example, to manage our time, optimize our diets, and achieve our fitness goals. To do so effectively, these apps need a predictive and mathematically tractable model of our behavior. The quasi-hyperbolic discounting model is promising: economists have shown in both lab and field experiments that it is highly predictive. However, the prior literature fails to provide a suitable framework in which to reason about quasi-hyperbolic discounting in the presence of complex planning tasks.

In the following paper, Kleinberg and Oren describe a graph-theoretic framework for task planning with quasi-hyperbolic discounting. In their framework, the goal and the intermediate tasks are nodes in a graph, and the weights of the (directed)

edges represent the costs of advancing from one task to the next. The intended present and future actions of a quasi-hyperbolic discounter are simply weighted shortest-paths in this graph.

This formulation allows researchers to use the extensive existing knowledge of graph algorithms to analyze and optimize task planning. In the paper, Kleinberg and Oren use their framework to derive many results. Among these, they characterize tasks that are particularly susceptible to procrastination: they are those that contain a simple structure as a graph minor. They also explore ways to reduce procrastination by choice reduction: by scheduling a midterm exam, an instructor can remove the choice of cramming for the final in the last week of class, resulting in better study habits.

The framework of Kleinberg and Oren, and the characterization and optimization results it enables, is a step forward in incorporating sophisticated models of human behavior into computational systems. Apps designed to increase individual effectiveness and related products can use this framework to help us achieve our personal goals. They can predict when we are in danger of procrastinating, and perhaps, by cleverly hiding the availability of certain actions, they can even help us mitigate the extent of procrastination. Subsequent and future work has and will continue to use the framework to develop many more such useful results. In our complex modern world with its growing plethora of choices, automated planning of the sort aided by this paper and the research it inspires is indispensable. 

The framework of Kleinberg and Oren is a step forward in incorporating sophisticated models of human behavior into computational systems.

To view the accompanying paper, visit doi.acm.org/10.1145/3176189 

Nicole Immorlica (nicimm@microsoft.com) is a senior researcher at Microsoft Research New England, Cambridge, MA, USA.

Copyright held by author.

Time-Inconsistent Planning: A Computational Problem in Behavioral Economics

By Jon Kleinberg and Sigal Oren

Abstract

In many settings, people exhibit behavior that is inconsistent across time—we allocate a block of time to get work done and then procrastinate, or put effort into a project and then later fail to complete it. An active line of research in behavioral economics and related fields has developed and analyzed models for this type of time-inconsistent behavior.

Here we propose a graph-theoretic model of tasks and goals, in which dependencies among actions are represented by a directed graph, and a time-inconsistent agent constructs a path through this graph. We first show how instances of this path-finding problem on different input graphs can reconstruct a wide range of qualitative phenomena observed in the literature on time-inconsistency, including procrastination, abandonment of long-range tasks, and the benefits of reduced sets of choices. We then explore a set of analyses that quantify over the set of all graphs; among other results, we find that in any graph, there can be only polynomially many distinct forms of time-inconsistent behavior; and any graph in which a time-inconsistent agent incurs significantly more cost than an optimal agent must contain a large “procrastination” structure as a minor. Finally, we use this graph-theoretic model to explore ways in which tasks can be designed to motivate agents to reach designated goals.

1. INTRODUCTION

A fundamental issue in behavioral economics—and in the modeling of individual decision-making more generally—is to understand the effects of decisions that are inconsistent over time. Examples of such inconsistency are widespread in everyday life: we make plans for completing a task but then procrastinate; we put work into getting a project partially done but then abandon it; we pay for gym memberships but then fail to make use of them. In addition to analyzing and modeling these effects, there has been increasing interest in incorporating them into the design of policies and incentives in domains that range from health to personal finance.

These types of situations have a recurring structure: a person makes a plan at a given point in time for something they will do in the future (finishing homework, exercising, paying off a loan), but at a later point in time they fail to follow through on the plan. Sometimes this failure is the result of unforeseen circumstances that render the plan invalid—a person might join a gym but then break their leg and be unable to exercise—but in many cases the plan is abandoned even though the circumstances are essentially the same as

they were at the moment the plan was made. This presents a challenge to any model of planning based on optimizing a utility function that is consistent over time: in an optimization framework, the plan must have been an optimal choice at the outset, but later it was optimal to abandon it. A line of work in the economics literature has thus investigated the properties of planning with objective functions that vary over time in certain natural and structured ways.

A basic example and model

To introduce these models, it is useful to briefly describe an example due to George Akerlof,¹ with the technical details adapted slightly for the discussion here. (The story will be familiar to readers who know Akerlof’s paper; we cover it in some detail because it will motivate a useful and recurring construction later in the work.) Imagine a decision-making agent—Akerlof himself, in his story—who needs to ship a package sometime during one of the next n days, labeled $t = 1, 2, \dots, n$, and must decide on which day t to do so. Each day that the package has not been sent results in a fixed cost of 1 (per day), due to the lack of use of the package’s contents; this means a cost of t if it is shipped on day t . (For simplicity, we will disregard the time the package spends in transit, which is a constant additional cost regardless of when it is shipped.) Also, shipping the package is an elaborate operation that will result in one-time cost of $c > 1$, due to the amount of time involved in getting it sent out. The package must be shipped during one of the n specified days.

What is the optimal plan for shipping the package? Clearly the cost c will be incurred exactly once regardless of the day on which it is shipped, and there will also be a cost of t if it is shipped on day t . Thus we are minimizing $c + t$ subject to $1 \leq t \leq n$; the cost is clearly minimized by setting $t = 1$. In other words, the agent should ship the package right away.

But in Akerlof’s story, he did something that should be familiar from one’s own everyday experience: he procrastinated. Although there were no unexpected changes to the trade-offs involved in shipping the package, when each new day arrived there seemed to be other things that were more crucial than sending it out that day, and so each day he resolved that he would instead send it tomorrow. The result was that the package was not sent out until the end of the

The original version of this paper was published in the *Proceedings of the 15th ACM Conference on Economics and Computation* (Palo Alto, CA, June 8–12, 2014), 547–564.

time period. (In fact, he sent it a few months into the time period once something unexpected happened to change the cost structure—a friend offered to send it as part of a larger shipment—though this wrinkle is not crucial for the story.)

There is a natural way to model an agent’s decision to procrastinate, using the notion of *present bias*—the tendency to view costs and benefits that are incurred at the present moment to be more salient than those incurred in the future. In particular, suppose that for a constant $b > 1$, costs that one must incur in the current time period are increased by a factor of b in one’s evaluation.^a Then in Akerlof’s example, when the agent on day t is considering the decision to send the package, the cost of sending it on day t is $bc + t$, while the cost of sending it on day $t + 1$ is $c + t + 1$. The difference between these two costs is $(b - 1)c - 1$, and so if $(b - 1)c > 1$, the agent will decide on each day t that the optimal plan is to wait until day $t + 1$; things will continue this way until day n , when waiting is no longer an option and the package must be sent.

Quasi-hyperbolic discounting

Building on considerations such as those above, and others in earlier work in economics,^{16,19} a significant amount of work has developed around a model of time-inconsistency known as *quasi-hyperbolic discounting*.¹² In this model, parametrized by quantities $\beta, \delta \leq 1$, a cost or reward of value c that will be realized at a point $t \geq 1$ time units into the future is evaluated as having a present value of $\beta^\delta c$. (In other words, values at time t are discounted by a factor of β^δ .) With $\beta = 1$ this is the standard functional form for exponential discounting, but when $\beta < 1$ the function captures present bias as well: values in the present time period are scaled up by β^{-1} relative to all other periods. (In what follows, we will consistently use b to denote β^{-1} .)

Research on this (β, δ) -model of discounting has been extensive, and has proceeded in a wide variety of directions; see Ref. Frederick et al.⁷ for a review. To keep our analysis clearly delineated in scope, we make certain decisions at the outset relative to the full range of possible research questions: we focus on a model of agents who are *naive*, in that they do not take their own time-inconsistency into account when planning; we do not attempt to derive the (β, δ) -model from more primitive assumptions but rather take it as a self-contained description of the agent’s observed behavior; and we discuss the case of $\delta = 1$ so as to focus attention on the present-bias parameter β . Note that the initial Akerlof example has all these properties; it is essentially described in terms of the (β, δ) -model with an agent who is naive about his own time-inconsistency, with $\delta = 1$, and with $\beta = b^{-1}$ (using the parameter b from that discussion).

Our starting point in this paper is to think about some of the qualitative predictions of the (β, δ) -model, and how to analyze them in a unified framework. In particular, research in behavioral economics has shown how agents making plans in this model can exhibit the following behaviors.

^a Note that there is no time-discounting in this example, so the factor of b is only applied to the present time period, while all future time periods are treated equally. We will return to the issue of discounting shortly.

1. *Procrastination*, as discussed above.
2. *Abandonment* of long-range tasks, in which a person starts on a multi-stage project but abandons it in the middle, even though the costs and benefits of the project have remained essentially unchanged.^{15,b}
3. The benefits of *choice reduction*, in which reducing the set of options available to an agent can actually help them reach a goal more efficiently.^{9, 14} A canonical example is the way in which imposing a deadline can help people complete a task that might not get finished in the absence of a deadline.⁴

These consequences of time-inconsistency, as well as a number of others, have in general each required their own separate and sometimes quite intricate modeling efforts. It is natural to ask whether there might instead be a single framework for representing tasks and goals in which all of these effects could instead emerge “mechanically,” each just as a different instance of the same generic computational problem. With such a framework, it would become possible to search for worst-case guarantees, by quantifying over all instances, and to talk about designing or modifying given task structures to induce certain desired behaviors.

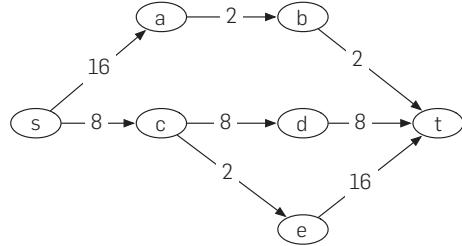
The present work: A graph-theoretic model

Here we propose such a framework, using a graph-theoretic formulation. We consider an agent with present-bias parameter β who must construct a path in a directed acyclic graph G with edge costs, from a designated start node s to a designated target node t . We assume without loss of generality that all the edges of G lie on some $s - t$ path. We will call such a structure a *task graph*. Informally, the nodes of the task graph represent states of intermediate progress toward the goal t , and the edges represent transitions between them. Directed graphs have been shown to have considerable expressive power for planning problems in the artificial intelligence literature;¹⁸ this provides evidence for the robustness of a graph-based approach in representing these types of decision environments. Our concerns in this work, however, are quite distinct from the set of graph-based planning problems in artificial intelligence, since our aim is to study the consequences of time-inconsistency in these domains.

A sample instance of this problem is depicted in Figure 1, with the costs drawn on the edges. When the agent is standing at a node v , it determines the minimum cost of a path from v to t , but it does so using its present-biased evaluation of costs: the cost of the first edge on the path (starting from v) is evaluated according to the true cost, and all subsequent edges have their costs reduced by β . If the agent chooses path P , it follows just the first edge (v, w) of P , and then it re-evaluates which path to follow using this same present-biased evaluation but now from w . In this way, the agent iteratively constructs a path from s to t .

^b For purposes of our discussion, we distinguish abandonment of a task from the type of procrastination exhibited by Akerlof’s example, in which the task is eventually finished, but at a much higher cost due to the effect of procrastination.

Figure 1. A present-biased agent must choose a path from s to t .



In the next section we will show how our graph-theoretic model easily captures time-inconsistency phenomena including procrastination, abandonment, and choice reduction. But to make the definitions concrete, it is useful to work through the agent's computation on the graph depicted in Figure 1. As shown in Figure 1, an agent that has a present-bias parameter of $\beta = 1/2$ needs to go from s to t . From s , the agent evaluates the path $s-a-b-t$ as having cost $16 + 2\beta + 2\beta = 18$, the path $s-c-d-t$ as having cost $8 + 8\beta + 8\beta = 16$, and the path $s-c-e-t$ as having cost $8 + 2\beta + 16\beta = 17$. Thus the agent traverses the edge (s, c) and ends up at c . From c , the agent now evaluates the path $c-d-t$ as having cost $8 + 8\beta = 12$ and the path $c-e-t$ as having cost $2 + 16\beta = 10$, and so the agent traverses the edge (c, e) and then (having no further choices) continues on the edge (e, t) .

This example illustrates a few points. First, when the agent set out on the edge (s, c) , it was intending to next follow the edge (c, d) , but when it got to c , it changed its mind and followed the edge (c, e) . A time-consistent agent (with $\beta = 1$), in contrast, would never do this; the path it decides to take starting at s is the path it will continue to follow all the way to t . Second, we are interested in whether the agent minimizes the cost of traveling from s to t according to the real costs, not according to its evaluation of the costs, and in this regard it fails to do so; the shortest path is $s-a-b-t$, with a cost of 20, while the agent incurs a cost of 26.

Overview of results

Our graph-theoretic framework makes it possible to reason about time-inconsistency effects that arise in very different settings, provided simply that the underlying decisions faced by the agent can be modeled as the search for a path through a graph-structured sequence of options. And perhaps more importantly, since it is tractable to ask questions that quantify over all possible graphs, we can cleanly compare different scenarios, and search for the best or worst possible structures relative to specific objectives. This is difficult to do without an underlying combinatorial structure. For example, suppose we were inspired by Akerlof's example to try identifying the scenario in which time-inconsistency leads to the greatest waste of effort. *A priori*, it is not clear how to formalize the search over all possible "scenarios." But as we will see, this is precisely something we can do if we simply ask for the graph in which time-inconsistency produces the greatest ratio between the cost of the path traversed and cost of the optimal path.

Moreover, with this framework in place, it becomes easier to express formal questions about design for these contexts: if as a designer of a complex task we are able to specify the underlying graph structure, which graphs will lead time-inconsistent agents to reach the goal efficiently?

Our core questions are based on quantifying the inefficiency from time-inconsistent behavior, designing task structures to reduce this inefficiency, and comparing the behavior of agents with different levels of time-inconsistency. Specifically, we ask:

1. In which graph structures does time-inconsistent planning have the potential to cause the greatest waste of effort relative to optimal planning?
2. How do agents with different levels of present bias (encoded as different values of β) follow combinatorially different paths through a graph toward the same goal?
3. Can we increase an agent's efficiency in reaching a goal by deleting nodes and/or edges from the underlying graph, thus reducing the number of options available?

In what follows, we address these questions in turn. For the first question, we consider n -node graphs and ask how large the *cost ratio* can be between the path followed by a present-biased agent and the path of minimum total cost. Since deviations from the minimum-cost plan due to present bias are sometimes viewed as a form of "irrational" behavior, this cost ratio effectively serves as a "price of irrationality" for our system. We give a characterization of the worst-case graphs in terms of *graph minors*; this enables us to show, roughly speaking, that any instance with sufficiently high cost ratio must contain a large instance of the Akerlof example embedded inside it.

For the second question, we consider the possible paths followed by agents with different present-bias parameters β . As we sweep β over the interval $[0, 1]$, we have a type of *parametric* path problem, where the choice of path is governed by a continuous parameter (β in this case). We show that in any instance, the number of distinct paths is bounded by a polynomial function of n , which forms an interesting contrast with canonical formulations of the parametric shortest-path problem, in which the number of distinct paths can be super polynomial in n .^{5,13}

Lastly, for the third question, we show how it is possible for agents to be more efficient when nodes and/or edges are deleted from the underlying graph; on the other hand, if we want to motivate an agent to follow a particular path P through the graph, it can be crucial to present the agent with a subgraph that includes not just P but also certain additional nodes and edges that do not belong to P . We give a graph-theoretic characterization of the possible subgraphs supporting efficient traversal.

Before turning to these questions, we first discuss the basic graph-theoretic problem in more detail, showing how instances of this problem capture the time-inconsistency phenomena discussed earlier in this section.

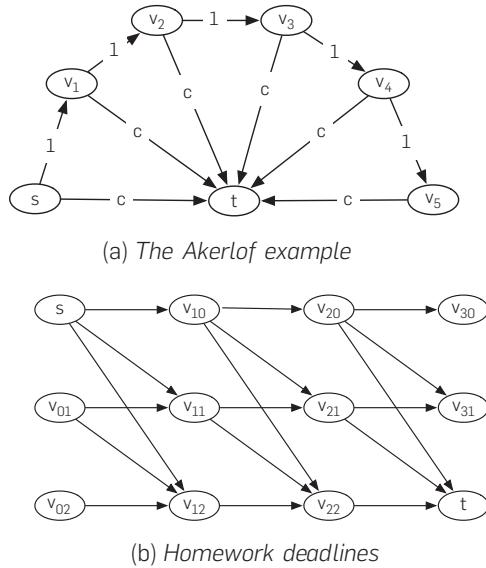
2. THE GRAPH-THEORETIC MODEL

In order to argue that our graph-theoretic model captures a variety of phenomena that have been studied in connection with time-inconsistency, we present a sequence of examples to illustrate some of the different behaviors that the model exhibits. We note that the example as shown in Figure 1 already illustrates two simple points: that the path chosen by the agent can be suboptimal; and that even if the agent traverses an edge e with the intention of following a path P that begins with e , it may end up following a different path P' that also begins with e .

For an edge e in G , let $c(e)$ denote the cost of e ; and for a path P in G , let $e_i(P)$ denote the i^{th} edge on P . In terms of this notation, the agent's decision is easy to specify: when standing at a node v , it chooses the path P that minimizes $c(e_1(P)) + \beta \sum_{i>1} c(e_i(P))$ over all P that run from v to t . It follows the first edge of P to a new node w , and then performs this computation again.

We begin by observing that Figure 2(a) represents a version of the Akerlof example from the introduction. (Recall that here we use b to denote β^{-1} .) Node t represents the state in which the agent has sent the package, and node v_i represents the state in which the agent has reached day i without sending the package. The agent has the option of going directly from node s to node t , and this is the shortest $s-t$ path. But if $(b-1)c > b$, then the agent will instead go from s to v_1 , intending to complete the path $s-v_1-t$ in the next time step. At v_1 , however, the agent decides to go to v_2 , intending to complete the path v_1-v_2-t in the next time step. This process continues: the agent, following exactly the reasoning in the example from the introduction, is procrastinating and not going to t , and in the end its path goes all the way to the last node v_n ($n = 5$ in the figure) before finally taking an edge to t . (One minor change from the set-up in the introduction is that the present-bias effect is also applied to the per-day cost of 1 as well; this has no real effect on the underlying story.)

Figure 2. Path problems that exhibit procrastination, abandonment, and choice reduction.



Extending the model to include rewards

Thus far we can not talk about an agent who abandons its pursuit of the goal midway through, since our model requires the agent to construct a path that goes all the way to t . A simple extension of the model enables to consider such situations.

Suppose we place a reward of r at the target node t , which will be claimed if the agent reaches t . Standing at a node v , the agent now has an expanded set of options: it can follow an edge out of v as before, or it can quit taking steps, incurring no further cost but also not claiming the reward. The agent will choose the latter option precisely when either there is no $v-t$ path, or when the minimum cost of a $v-t$ path exceeds the value of the reward, evaluated in light of present bias: $c(e_1(P)) + \beta \sum_{i>1} c(e_i(P)) > \beta r$ for all $v-t$ paths P . It is important to note a key feature of this evaluation: the reward is always discounted by β relative to the cost that is being incurred in the current period, even if the reward will be received right after this cost is incurred. (e.g., if the path P has a single edge, then the agent is comparing $c(e_1(P))$ to βr .)

In what follows, we will consider both these models: the former *fixed-goal model*, in which the agent must reach t and seeks to minimize its cost; and the latter *reward model* in which the agent trades off cost incurred against reward at t , and has the option of stopping partway to t . Aside from this distinction, both models share the remaining ingredients, based on traversing an $s-t$ path in G .

It is easy to see that the reward model displays the phenomenon of *abandonment*, in which the agent spends some cost to try reaching t , but then subsequently gives up without receiving the reward. Consider for example a three-node path on nodes s , v_1 , and t , with an edge $c(s, v_1) = 1$ and $c(v_1, t) = 4$. If $\beta = 1/2$ and there is a reward of 7 at t , then the agent will traverse the edge (s, v_1) because it evaluates the total cost of the path at $1 + 4\beta = 3 < 7\beta = 3.5$. But once it reaches v_1 , it evaluates the cost of completing the path at $4 > 7\beta = 3.5$, and so it quits without reaching t .

An example involving choice reduction

It is useful to describe a more complex example that shows the modeling power of this shortest-path formalism, and also shows how we can use the model to analyze deadlines as a form of beneficial choice reduction. (As should be clear, with a time-consistent agent it can never help to reduce the set of choices; such a phenomenon requires some form of time-inconsistency.) First we describe the example in text, and then show how to represent it as a graph.

Imagine a student taking a three-week short course in which the required work is to complete two small projects by the end of the course. It is up to the student when to do the projects, as long as they are done by the end. The student incurs an effort cost of one from any week in which she does no projects (since even without projects there is still the lower-level effort of attending class), a cost of four from any week in which she does one project, and a cost of nine from any week in which she does both projects. Finally, the student receives a reward of 16 for completing the course, and she has a present-bias parameter of $\beta = 1/2$.

Figure 2(b) shows how to represent this scenario using a graph. Node v_{ij} corresponds to a state in which i weeks of the course are finished, and the student has completed j projects so far; we have $s = v_{00}$ and $t = v_{32}$. All edges go one column to the right, indicating that one week will elapse regardless; what is under the student's control is how many rows the edge will span. Horizontal edges have cost one, edges that descend one row have cost four, and edges that descend two rows in a single hop have cost nine. In this way, the graph precisely represents the story just described.

How does the student's construction of an $s-t$ path work out? From s , she goes to v_{10} and then to v_{20} , intending to complete the path to t via the edge (v_{20}, t) . But at v_{20} , she evaluates the cost of the edge (v_{20}, t) as $9 > \beta r = 16/2 = 8$, and so she quits without reaching t . The story is thus a familiar one: the student plans to do both projects in the final week of the course, but when she reaches the final week, she concludes that it would be too costly and so she drops the course instead.

The instructor can prevent this from happening through a very simple intervention. If he requires that the first project be completed by the end of the second week of the course, this corresponds simply to deleting node v_{20} from the graph. With v_{20} gone, the path-finding problem changes: now the student starting at s decides to follow the path $s-v_{10}-v_{21}-t$, and at v_{10} and then v_{21} she continues to select this path, thereby reaching t . Thus, by reducing the set of options available to the student—and in particular, by imposing an intermediate deadline to enforce progress—the instructor is able to induce the student to complete the course.

There are many stories like this one about homework and deadlines, and our point is not to focus too closely on it in particular. Indeed, to return to one of the underpinnings of our graph-theoretic formalism, our point is in a sense the opposite: it is hard to reason about the space of possible “stories,” whereas it is much more tractable to think about the space of possible graphs. Thus by encoding the set of stories mechanically in the form of graphs, it becomes feasible to reason about them as a whole.

We have thus seen how a number of different time-inconsistency phenomena arise in simple instances of the path-finding problem. The full power of the model, however, lies in proving statements that quantify over all graphs; we begin this next.

3. THE COST RATIO: A CHARACTERIZATION VIA GRAPH MINORS

Our path-finding model naturally motivates a basic quantity of interest: the *cost ratio*, defined as the ratio between the cost of the path found by the agent and the cost of the shortest path. We work here within the fixed-goal version of the model, in which the agent is required to reach the goal t and the objective is to minimize the cost of the path used.

To fix notation for this discussion, given a directed acyclic graph G on n nodes with positive edge costs, we let $d(v, w)$ denote the cost of the shortest $v-w$ path in G (using the true edge costs, not modified by present bias). Let $P_\beta(v, t)$ denote the $v-t$ path followed by an agent with present-bias β , and let $c_\beta(v, t)$ be the total cost of this path. The cost ratio can thus be written as $c_\beta(s, t)/d(s, t)$.

A bad example for the cost ratio

We first describe a simple construction showing that the cost ratio can be exponential in the number of nodes n . We then move on to the main result of this section, which is a characterization of the instances in which the cost ratio achieves this exponential lower bound.

Our construction is an adaptation of the Akerlof example from the introduction. We have a graph that consists of a directed path $s = v_0, v_1, v_2, \dots, v_n$, and with each v_i also linking directly to node t . (The case $n = 5$ is the graph in Figure 2(a).) With $b = \beta^{-1}$, we choose any $\mu < b$; we let the cost of the edge (v_j, t) be μ^j , and let the cost of each edge (v_j, v_{j+1}) be 0.

Now, when the agent is standing at node v_j , it evaluates the cost of going directly to t as μ^j , while the cost of the two-step path through v_{j+1} to t is evaluated as $0 + \beta\mu^{j+1} = (\beta\mu)\mu^j < \mu^j$. Thus the agent will follow the edge (v_j, v_{j+1}) with the plan of continuing from v_{j+1} to t . But this holds for all j , so once it reaches v_{j+1} , it changes its mind and continues on to v_{j+2} , and so forth. Ultimately it reaches v_n , and then must go directly to t at a cost of $c_\beta(s, t) = \mu^n$. Since $d(s, t) = 1$ by using the edge directly from s to t , this establishes the exponential lower bound on the cost ratio $c_\beta(s, t)/d(s, t)$. Essentially, this construction shows that the Akerlof example can be made quantitatively much worse than its original formulation by having the cost of going directly to the goal grow by a modest constant factor in each time step; when a present-biased agent procrastinates in this case, it ultimately incurs an exponentially large cost.

The following observation establishes this is the highest possible cost ratio

OBSERVATION 3.1. Consider an agent currently at v and let u be the next node on $P_\beta(v, t)$. Then $d(u, t) < b \cdot d(v, t)$.

PROOF. If u is on the shortest path from v to t then clearly the claim holds. Else, since the agent chose to continue to u instead of continuing on the shortest path from v to t we have $c(v, u) + \beta d(u, t) \leq d(v, t)$. This implies that $d(u, t) \leq b \cdot d(v, t)$. \square

The observation essentially implies that with each step that the agent takes the cost of the path that it plans to take increases by an extra factor of at most b relatively to $d(s, t)$. Hence, we have a tight upper bound on the cost ratio:

COROLLARY 3.2. The cost ratio for a graph G with n nodes is at most b^{n-2} .

A graph minor characterization

We now provide a structural description of the graphs on which the cost ratio can be exponential in the number of nodes n —essentially we show that a constant fraction of the nodes in such a graph must have the structure of the Akerlof example.

We make this precise using the notion of a *minor*. Given two undirected graphs H and K , we say that H contains a K -minor if we can map each node κ of K to a connected subgraph S_κ in H , with the properties that (i) S_κ and $S_{\kappa'}$ are disjoint for every two nodes κ, κ' of K , and (ii) if (κ, κ') is an edge of K , then in H there is some edge connecting a node in S_κ to

a node in S_{κ} . Informally, the definition means that we can build a copy of K using the structure of H , with disjoint connected subgraphs of H playing the role of “super-nodes” that represent the nodes of K , and with the adjacencies among these super-nodes representing the adjacencies in K . The minor relation shows up in many well-known results in graph theory, perhaps most notably in Kuratowski’s Theorem that a nonplanar graph must contain either the complete graph K_5 or the complete bipartite graph $K_{3,3}$ as a minor.⁶

Our goal here is to show that if G has exponential cost ratio, then its undirected version must contain a large copy of the graph underlying the Akerlof example as a minor. In other words, the Akerlof example is not only a way to produce a large cost ratio, but it is in a sense an unavoidable signature of any example in which the cost ratio is very large.

We set this up as follows. Let $\sigma(G)$ denote the skeleton of G , the undirected graph obtained by removing the directions on the edges of G . Let \mathcal{F}_k denote the graph with nodes v_1, v_2, \dots, v_k , and w , and edges (v_i, v_{i+1}) for $i = 1, \dots, k-1$, and (v_i, w) for $i = 1, \dots, k$. We refer to \mathcal{F}_k as the k -fan.

We now claim

THEOREM 3.3. *For every $\lambda > 1$, if n is sufficiently large and the cost ratio is greater than λ^n , then $\sigma(G)$ contains an \mathcal{F}_k -minor for $k = \Theta(n)$.*

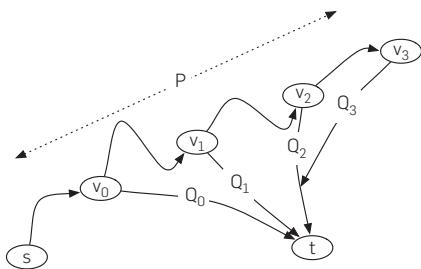
Proof sketch.^c We now provide a sketch of the proof. The basic idea is to pin down $k = \Theta(n)$ nodes, v_1, \dots, v_k , on the path $P_\beta(s, t)$ and let P be the portion of the path from s to v_k . We show that from each v_i the shortest path Q_i intersects with P only at v_i . This is illustrated in Figure 3.

Once we have this structure, we obtain an \mathcal{F}_k minor by partitioning P into segments such that each v_i is in a distinct segment, and then we contract each segment. Since all the Q_i ’s are connected to t , we can contract them together (without the v_i ’s). Finally, as all segments of P are connected to one another, and each segment is connected to t by Q_i , we get a fan \mathcal{F}_k .

For simplicity we normalize the edges costs such that $d(s, t) = 1$. We choose the nodes v_1, \dots, v_k such that for every i , v_i is the last node on the path P such that $d(v_i, t) \leq b^i$. With this choice it is not hard to show that the shortest path from v_i to t (Q_i) intersects with P only at v_i . In particular, Q_i can not intersect with P at any node before v_i since this will create

^c This sketch is based on the full proof in our original paper, and also draws on the exposition in Tim Roughgarden’s lecture notes about our paper.¹⁷

Figure 3. The construction of an \mathcal{F}_k -minor in Theorem 3.3.



a directed cycle and our graph is a directed acyclic graph. Furthermore, Q_i can not intersect with P at any node u after v_i since by our choice of v_i , we have $d(u, t) > b^i \geq d(v_i, t)$ for every node u after v_i .

Finally, we should show that indeed for every $1 \leq i \leq k$, there exists a distinct node v_i with the property that (i) $d(v_i, t) \leq b^i$, and (ii) for any node u after v_i on $P_\beta(s, t)$, we have $d(u, t) > b^i$. First observe that since the cost ratio is λ^n and the length of the path is at most n , the path must contain an edge (u, v) of cost at least λ^n/n . Roughly speaking, since n is large enough there exists $\lambda_0 > 1$ such that $\lambda^n/n = \lambda_0^n$. In particular this implies that $d(u, t) \geq \lambda_0^n$. By Observation 3.1 we have that with each step on the path P the cost of the shortest path can increase by at most a factor of b . Thus, there exist k nodes as required.

After the initial publication of our original paper, Tang et al.²⁰ extended Theorem 3.3 as follows:

THEOREM 3.4. *If the cost ratio is greater than b^{k-2} , then $\sigma(G)$ contains an \mathcal{F}_k -minor.*

Both Theorem 3.3 and its tighter counterpart Theorem 3.4 offer some qualitatively relevant advice for thinking about the structure of complicated tasks: to avoid creating inefficient behavior due to present bias, it is better to organize tasks so that they do not contain large fan-like structures. The point to appreciate is that such fan-like structures are not purely graph-theoretic abstractions; they arise in real settings whenever a task has a series of “branches” (as in Akerlof’s story) that allows an agent to repeatedly put off completing the task. The theorems are a way of formalizing the idea that such sets of repeated branches are the crux of the reason why present-biased individuals incur unnecessary inefficiency in completing large tasks. And correspondingly, organizing tasks in a way that breaks this type of branching—for example, with intermediate deadlines or subgoals—can be a way of reducing inefficiency.

4. COLLECTIONS OF HETEROGENEOUS AGENTS

Thus far we have focused on the behavior of a single agent with a given present-bias parameter β . Now we consider all possible values of β , and ask the following basic question: how large can the set $\{P_\beta(s, t) : \beta \in [0, 1]\}$ be? In other words, if for each β , an agent with parameter β were to construct an $s-t$ path in G , how many different paths would be constructed across all the agents? Bounding this quantity tells us how many genuinely “distinct” types of behaviors there are for the instance defined by G . Let $\mathcal{P}(G)$ denote the set $\{P_\beta(s, t) : \beta \in [0, 1]\}$. Despite the fact that β comes from the continuum $[0, 1]$, the set $\mathcal{P}(G)$ is clearly finite, since G only has finitely many $s-t$ paths. The question is whether we can obtain a nontrivial upper bound on the size of $\mathcal{P}(G)$, and in particular one that does not grow exponentially in the number of nodes n . In fact this is possible, and our main goal in this section is to prove the following theorem.

THEOREM 4.1. *For every directed acyclic graph G , the size of*

$\mathcal{P}(G)$ is $O(n^2)$. Moreover, there exists a graph for which the size of $\mathcal{P}(G)$ is $\Omega(n^2)$.

Proof idea. We use the following procedure to “discover” all the paths in $\mathcal{P}(G)$. We start by taking $\beta = 0$ and let P_0 the path that the agent with $\beta = 0$ takes. Now, we gradually increase β till we reach β^* such that an agent with β^* will take a different path. We claim that there is at least one edge (v, u) in P_0 that will not take part in any path that an agent with $\beta > \beta^*$ will take. More generally, each time that we discover a new path, we essentially delete at least one edge from the graph. Hence the number of paths in $\mathcal{P}(G)$ is bounded by the number of edges in the graph.

Theorem 4.1 tells us that the effect of present-bias on the path that an agent takes is in some sense limited as the specific value of β an agent has determines which path will the agent take from a precomputed set of at most $O(n^2)$ different paths. Since this $O(n^2)$ quantity is much smaller in general than the full set of possible paths, it says that the possible heterogeneity in agent behavior based on different levels of present bias is not as extensive as it might initially seem. Furthermore, quantifying this heterogeneity is a first step in designing efficient task graphs for populations of agents that are heterogeneous.

5. MOTIVATING AN AGENT TO REACH THE GOAL

We now consider the version of the model with rewards: there is a reward at t , and the agent has the additional option of quitting if it perceives—under its present-biased evaluation—that the value of the reward is not worth the remaining cost in the path.

Note that the presence of the reward does not affect the agent’s choice of path, only whether it continues along the path. Thus we can clearly determine the minimum reward r required to motivate the agent to reach the goal in G by simply having it construct a path to t according to our standard fixed-goal model, identifying the node at which it perceives the remaining cost to be the greatest (due to present bias this might not be s), and assigning this maximum perceived cost as a reward at t .

A more challenging question is suggested by the possibility of deleting nodes and edges from G ; recall that Figure 2(b) showed a basic example in which the instructor of a course was able to motivate a student to finish the coursework by deleting a node from the underlying graph. (This deletion essentially corresponded to introducing a deadline for the first piece of work.) This shows that even if the reward remains fixed, in general it may be possible for a designer to remove parts of the graph, thereby reducing the set of options available to the agent, so as to get the agent to reach the goal. We now consider the structure of the subgraphs that naturally arise from this process.

Motivating subgraphs: A fundamental example

The basic set-up we consider is the following. Suppose the agent in the reward model is trying to construct a path from s to t in G ; the reward r is not under our control—perhaps it is defined by a third party, or represents an intrinsic reward that we can not augment—but we are able to remove nodes and edges from the graph (essentially by declaring certain

activities invalid, as the deadline did in Figure 2(b)). Let us say that a subgraph G' of G motivates the agent if in G' with reward r , the agent reaches the goal node t . (We will also refer to G' as a *motivating subgraph*.) Note that it is possible for the full graph G to be a motivating subgraph.

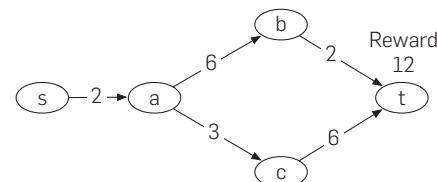
It would be natural to conjecture that if there is any subgraph G' of G that motivates the agent, then there is a motivating subgraph consisting simply of an s - t path P that the agent follows. In fact this is not the case. Figure 4 shows a graph illustrating a phenomenon that we find somewhat surprising *a priori*, though not hard to verify from the example. In the graph G depicted in the figure, an agent with $\beta = 1/2$ will reach the goal t . However, there is no proper subgraph of G in which the agent will reach the goal. The point is that the agent starts out expecting to follow the path s - a - b - t , but when it gets to node a it finds the remainder of the path a - b - t too expensive to justify the reward, and it switches to a - c - t for remainder. With just the path s - a - b - t in isolation, the agent would get stuck at a ; and with just s - a - c - t , the agent would never start out from s . It is crucial the agent mistakenly believes the upper path is an option in order to eventually use the lower path to reach the goal.

It is interesting, of course, to consider real-life analogues of this phenomenon. In some settings, the structure in Figure 4 could correspond to deceptive practices on the part of the designer of G —in other words, inducing the agent to reach the goal by misleading them at the outset. But there are other settings in real life where one could argue that the type of deception represented here is more subtle, not any one party’s responsibility, and potentially even salutary. For example, suppose the graph schematically represents the learning of a skill such as a musical instrument. There’s the initial commitment corresponding to the edge (s, a) , and then the fork at a where one needs to decide whether to “get serious about it” (taking the expensive edge (a, b)) or not (taking the cheaper edge (a, c)). In this case, the agent’s trajectory could describe the story of someone who derived personal value from learning the violin (the lower path) even though at the outset they believed incorrectly that they would be willing to put the work into becoming a concert violinist (the upper path).

The structure of minimal motivating subgraphs

Given that there is sometimes no single path in G that is motivating, how rich a subgraph do we necessarily need to motivate the agent? Let us say that a subgraph G^* of G is a *minimal motivating subgraph* if (i) G^* is motivating, and (ii) no proper subgraph of G^* is motivating. Thus, for example,

Figure 4. A minimal subgraph for getting an agent to reach t .



in Figure 4, the graph G is a minimal motivating subgraph of itself; no proper subgraph of G is motivating.

Concretely, then, we can ask the following question: what can a minimal motivating subgraph look like? For example, could it be arbitrarily dense with edges?

In fact, minimal motivating subgraphs necessarily have a sparse structure, which we now describe in our next theorem. To set up this result, we need the following definition. Given a directed acyclic graph G and a path P in G , we say that a path Q in G is a P -bypass if the first and last nodes of Q lie on P , and no other nodes of Q do; in other words, $P \cap Q$ is equal to the two ends of Q . We now have

THEOREM 5.1. *If G^* is a minimal motivating subgraph, then it contains an s - t path P^* with the properties that*

- (i) *Every edge of G^* is either part of P^* or lies on a P^* -bypass in G^* ; and*
- (ii) *Every node of G^* has at most one outgoing edge that does not lie on P^* .*

Proof sketch. Roughly speaking, there are two types of edges that should be included in a minimal motivating subgraph: edges that the agent will actually take (these are the edges of the path P^*) and edges that at some point the agent (wrongly) believes that it will take (these are the edges on the P^* -bypasses). It is clearly the case that an edge e that the agent never plans to take can be safely removed from the graph without affecting the agent's decisions. Furthermore, since all the bypass edges are only used in shortest path computations it is impossible for a node v in P^* to have two neighbors w_1 and w_2 not on P^* such that the agent at some v_1 on P^* plans to follow a path that includes the edge (v, w_1) and an agent at some v_2 on P^* plans to follow a path that includes the edge (v, w_2) . This is simply because if $(v, w_1) + d(w_1, t) \leq (v, w_2) + d(w_2, t)$ the agent will choose the path that contains (v, w_1) both when standing at v_1 and at v_2 and otherwise it will choose the path that contains (v, w_2) in both cases.

After the publication of our original paper, Tang et al.²⁰ and Albers and Kraft² independently showed that determining whether a task graph admits a motivating subgraph is NP-complete. One way of circumventing these hardness results is to identify task graphs that are more common in practice and asking whether on these graphs the motivating subgraph problem can be solved in polynomial time. Alternatively, we can consider approximation algorithms. Albers and Kraft² considered a variant of this question: what is the minimum reward r for which a motivating subgraph exists? They showed that this problem can not be approximated by a factor better than $\sqrt{n}/3$, and they presented a $(\sqrt{n}+1)$ approximation algorithm. Interestingly, the subgraph achieving the $\sqrt{n}+1$ approximation is a path. Surprisingly, in a different paper, Albers and Kraft³ were able to break the \sqrt{n} barrier and presented a 2-approximation algorithm for a more powerful designer who is able to increase the costs of edges.

An alternative way to motivate an agent to reach t is to place intermediate rewards on specific nodes or edges; the agent will claim each reward if it reaches the node or edge

on which it is placed. Now the question is to place rewards on the nodes or edges of an instance G such that the agent reaches the goal t while claiming as little total reward as possible; this corresponds to the designer's objective to pay out as little as possible while still motivating the agent to reach the goal. Following our paper, Albers and Kraft² and Tang et al.²⁰ studied different versions of this question and showed that the problem of assigning intermediate rewards in an optimal way is NP-complete. A version worth mentioning is the one in which the designer only cares about minimizing the rewards that the agent actually takes. Such a formulation of the problem comes with the danger that a designer would be able to create "exploitive" solutions in which the agent is motivated by intermediate rewards that it will never claim, because these rewards are on nodes that the agent will never reach.

6. CONCLUSION AND SUBSEQUENT WORK

We have developed a graph-theoretic model in which an agent constructs a path from a start node s to a goal node t in an underlying graph G representing a sequence of tasks. Time-inconsistent agents may plan an s - t path that is different from the one they actually follow, and this type of behavior in the model can reproduce a range of qualitative phenomena including procrastination, abandonment of long-range tasks, and the benefits of a reduced set of options. Our results provide characterizations for a set of basic structures in this model, including for graphs achieving the highest cost ratios between time-inconsistent agents and shortest paths, and we have investigated the structure of minimal graphs on which an agent is motivated to reach the goal node.

There is a wide range of broader issues for further work. These include finding structural properties beyond our graph-minor characterization that have a bearing on the cost ratio of a given instance; obtaining a deeper understanding of the relationship between agents with different levels of time-inconsistency as measured by different values of β ; and developing algorithms for designing graph structures that motivate effort as efficiently as possible, including for multiple agents with diverse time-inconsistency properties.

In work following the initial appearance of our paper, the results were extended in several subsequent lines of research. We have discussed several of these further results in the text thus far, including a tighter graph-minor characterization of instances with high cost ratio,²⁰ and a set of hardness results and approximation algorithms for motivating subgraphs.^{2, 3, 20} Further results beyond these have considered the behavior of different types of agents. Gravin et al.⁸ studied the behavior of a present-biased agent whose present-bias parameter is not fixed over time; rather, after each step it re-samples the parameter from some fixed distribution. In Kleinberg et al.,¹⁰ the authors of the present paper together with Manish Raghavan studied the behavior of *sophisticated* agents (building on a formalization from O'Donoghue and Rabin¹⁴), who are aware of their present bias and can take it into account in their planning. Such agents are not able to ignore or disregard their present

bias—they still prefer avoiding costs in the present time period—but their sophistication does mean that they can take measures to reduce the future effects of present bias in their plans. Finally Kleinberg et al.,¹¹ studied the behavior of agents that exhibit two biases concurrently: present bias, as in this paper, and *sunk-cost bias*, which is the tendency to reason about costs already incurred in the formulation of plans for the future.

Acknowledgments

We thank Supreet Kaur, Sendhil Mullainathan, and Ted O'Donoghue for valuable discussions and suggestions. □

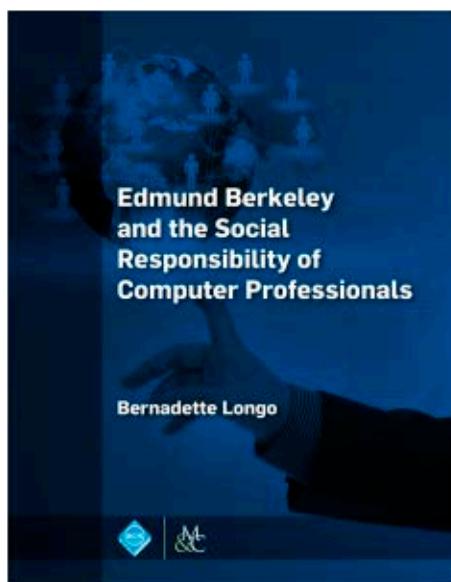
References

1. Akerlof, G.A. Procrastination and obedience. *American Econ. Rev.: Papers and Proc.* 81 (1991).
2. Albers, S., Kraft, D. Motivating time-inconsistent agents: A computational approach. In *Intl. Conf. Web and Internet Econ.* (2016).
3. Albers, S., Kraft, D. On the value of penalties in time-inconsistent planning. In *Proc. 44th Intl. Colloq. on Automata, Languages and Programming* (2017).
4. Ariely, D., Wertenbroch, K. Procrastination, deadlines, and performance: self-control by precommitment. *Psych. Sci.* 13 (2002).
5. Carstensen, P. The complexity of some problems in parametric linear and combinatorial programming. PhD thesis, U. Michigan (1983).
6. Diestel, R. *Graph Theory*, 3 edn. Springer (Berlin/Heidelberg, 2005).
7. Frederick, S., Loewenstein, G., O'Donoghue, T. Time discounting and time preference. *J. Econ. Lit.* 40, 2 (June 2002), 351–401.
8. Gravin, N., Immorlica, N., Lucier, B., Pountourakis, E. Procrastination with variable present bias. In *Proc. ACM Conf. Econ. Comp.* (2016).
9. Kaur, S., Kremer, M., Mullainathan, S. Self-control and the development of work arrangements. *American Econ. Rev.: Papers and Proceedings* 100 (2002).
10. Kleinberg, J., Oren, S., Raghavan, M. Planning problems for sophisticated agents with present bias. In *Proc. ACM Conf. Econ. Comp.* (2016).
11. Kleinberg, J., Oren, S., Raghavan, M. Planning with multiple biases. In *Proc. ACM Conf. Econ. Comp.* (2017).
12. Laibson, D. Golden eggs and hyperbolic discounting. *Q. J. Econ.* 112, 2 (1997), 443–478.
13. Nikolova, E., Kelner, J.A., Brand, M., Mitzenmacher, M. Stochastic shortest paths via quasi-convex maximization. In *Proc. 14th European Symposium on Algorithms* (2006), 552–563.
14. O'Donoghue, T., Rabin, M. Doing it now or later. *American Econ. Rev.* 89 (1999).
15. O'Donoghue, T., Rabin, M. Procrastination on long-term projects. *J. Econ. Behav. Org.* 66 (2008).
16. Pollak, R.A. Consistent planning. *Rev. Econ. Studies* 35, 2 (Apr. 1968), 201–208.
17. Roughgarden, T. Lecture 19: Time-inconsistent planning, 2016. <http://theory.stanford.edu/~tim/f16/L119.pdf>.
18. Russell, S.L., Norvig, P. *Artificial Intelligence: A Modern Approach*. Prentice Hall (Upper Saddle River NJ, USA, 1994).
19. Strotz, R.H. Myopia and inconsistency in dynamic utility maximization. *Rev. Econ. Studies* 23 (1955).
20. Tang, P., Teng, Y., Wang, Z., Xiao, S., Xu, Y. Computational issues in time-inconsistent planning. In *Proc. 31st AAAI Conf. Artificial Intelligence* (2017).

Jon Kleinberg (kleinber@cs.cornell.edu), Cornell University, Ithaca, NY, USA.

Sigal Oren (sigal3@gmail.com), Ben Gurion University of the Negev, Be'er Sheva, Israel.

© 2018 ACM 0001-0782/18/3 \$15.00



He was the conscience of the computing industry...and paid for it.

The first full-length biography of Edmund Berkeley, computing pioneer, social activist, and founding member of the ACM. It is an historical narrative of a man ultimately in favor of engineering peace, instead of war, and how his career was ultimately damaged by politicians determined to portray him as a Communist sympathizer. Berkeley's life work provides a lens to understand social and political issues surrounding the early development of electronic computers which ties directly to current debates about the use of autonomous intelligent systems.



ISBN: 978-1-970001-36-5 DOI: 10.1145/2787754
<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/berkeley>

Technical Perspective

On Heartbleed: A Hard Beginning Maketh a Good Ending

JOHN HEYWOOD (1497–1580)

By Kenny Paterson

THE SSL/TLS PROTOCOL suite has become the de facto secure protocol for communications on the Web, protecting billions of communications sessions between browsers and servers on a daily basis. We use it every time we access our social media feeds, or whenever an app running on our mobile device wants to contact its home server. It has become an almost invisible part of the Web's security infrastructure, supported by an eclectic mix of technologies including public key cryptography, certificates, and the Web PKI.

So when a serious security vulnerability is discovered in the SSL/TLS protocol itself, or in one of the main implementations like OpenSSL, one would naturally expect a rapid response—system administrators would roll into action, patching their software as quickly as possible, and taking any other remedial actions that might be necessary.

The following paper by Zhang et al. paints a very different picture in the context of the most famous SSL/TLS vulnerability of all, Heartbleed. The Heartbleed vulnerability resides in the OpenSSL implementation of the Heartbeat protocol. The Heartbeat protocol is an extension of SSL/TLS for checking the “liveness” of a connection. The Heartbleed bug lay in OpenSSL’s failure to correctly perform bounds checking when processing Heartbeat messages. An attacker, situated anywhere on the planet, could induce a server to return large amount of data to the attacker from arbitrary (but uncontrolled) portions of its stack. This memory leak would allow the attacker to learn a vulnerable server’s private key, with disastrous security consequences.

Heartbleed became public in early April 2014. The Internet community rapidly developed free Heartbleed scanning services, and published statistics on vulnerable servers. Some websites were actually attacked, though all hosts in the Alexa top 500 were patched within 48 hours. As well as immediately patching OpenSSL to remove the vulnerability, security experts recommended that sys-

tem administrators should revoke their public key certificates, generate new key pairs, and request their Certification Authorities (CAs) to issue new certificates. Private keys, among the security “crown jewels” for SSL/TLS, had potentially been compromised.

The following paper examines to what extent revocation and reissuance of certificates happened post-Heartbleed. The short and surprising answer: not so much. Zhang et al. cleverly use the Heartbleed incident as an opportunity to perform a natural experiment, tracking the rate at which large websites from the Alexa top 1M chose to revoke and reissue certificates. They carefully assess the extent to which those sites would have been vulnerable to Heartbleed, and then use the open nature of the Web to collect information about when those sites’ certificates were actually changed, and whether the previous certificates were properly revoked. To a security-conscious reader, the final statistics make depressing reading. For example, of roughly 107,712 vulnerable websites (that is, websites for which the private key could have been exposed), only 26.7% had reissued certificates by the end of April 2014, while 60% of those sites did not properly revoke their old certificates (meaning that, had the corresponding private keys been exposed, the sites would still be vulnerable).

The authors discuss some of the reasons why such low rates of revocation and reissuance were seen. They also report anecdotal evidence gleaned from surveying system administrators. A key reason is the processes are manual rather than being automated. One may also advance the argument that customers pay CAs for certificates, and security budgets are always under pressure. One possible perception is that the window of exposure was small, because most vulnerable systems were patched quickly. This belies the fact that the vulnerability was present in the OpenSSL code for

more than two years, and could have been exploited during that time. A third possible reason is ignorance on the part of sysadmins: while patching is part of the everyday sysadmin culture, dealing with certificates and keys is not. The paper points out a more systematic study is needed in order to understand certificate management and how it is tackled by sysadmins.

More broadly, the Heartbleed incident has had a lasting and net-positive impact on the security of the Web. The bug led to a much closer inspection of the OpenSSL code base, leading to other problems being subsequently discovered. It also led to a wider debate about the wisdom of having a security monoculture, about the quality of the OpenSSL code, and about the way in which the OpenSSL project was being run. And it triggered a debate about the responsibilities of large companies who make free use of open source software like OpenSSL without contributing materially to its development.

Heartbleed resulted in major industry players forming the Core Infrastructure Initiative, a project intended to fund critical elements of the global information infrastructure. OpenSSL has in turn significantly revised its operations, expanded the development team, and heavily refactored the codebase.

In parallel, Google initiated its own fork of OpenSSL called BoringSSL. The naming is considered, with “boring” being intended to convey an impression of “no nasty surprises.” In October 2015, Google announced it had switched over to BoringSSL across its entire codebase (comprising several billion lines of code).

The “Let’s Encrypt” initiative has started to address the problem of helping sysadmins to better manage certificates. Let’s Encrypt is a free CA, which simplifies the process of obtaining certificates and switching on SSL/TLS for websites. In mid-2017, its 100-millionth certificate was issued, and Mozilla’s telemetry indicated more than half of all HTTP connections were protected. □

Kenny Paterson is a professor of Information Security at Royal Holloway, University of London, U.K.

Copyright held by author.

Analysis of SSL Certificate Reissues and Revocations in the Wake of Heartbleed

By Liang Zhang, David Choffnes, Tudor Dumitras, Dave Levin, Alan Mislove, Aaron Schulman, and Christo Wilson

Abstract

A properly managed public key infrastructure (PKI) is critical to ensure secure communication on the Internet. Surprisingly, some of the most important administrative steps—in particular, reissuing new X.509 certificates and revoking old ones—are manual and remained unstudied, largely because it is difficult to measure these manual processes at scale.

We use Heartbleed, a widespread OpenSSL vulnerability from 2014, as a natural experiment to determine whether administrators are properly managing their certificates. All domains affected by Heartbleed should have patched their software, revoked their old (possibly compromised) certificates, and reissued new ones, all as quickly as possible. We find the reality to be far from the ideal: over 73% of vulnerable certificates were not reissued and over 87% were not revoked three weeks after Heartbleed was disclosed. Our results also show a drastic decline in revocations on the weekends, even immediately following the Heartbleed announcement. These results are an important step in understanding the manual processes on which users rely for secure, authenticated communication.

1. INTRODUCTION

Server authentication is the cornerstone of secure communication on the Internet; it is the property that allows client applications such as online banking, email, and e-commerce to ensure the servers with whom they communicate are truly who they say they are. In practice, server authentication is made possible by the globally distributed Public Key Infrastructure (PKI). The PKI leverages cryptographic mechanisms and X.509 certificates to establish the identities of popular websites. This mechanism works in conjunction with other network protocols—particularly Secure Sockets Layer (SSL) and Transport Layer Security (TLS)—to provide secure communications, but the PKI plays a key role: without it, a browser could establish a secure connection with an attacker that impersonates a trusted website.

The secure operation of the web's PKI relies on responsible administration. When a software vulnerability is discovered, administrators must act quickly and deploy the *patch* to prevent attackers from exploiting the vulnerability. Similarly, after a potential key compromise, website administrators must *revoke* the corresponding certificates to prevent attackers from intercepting encrypted communications

between browsers and servers. A recent study suggests 0.2% of SSL connections to Facebook correspond to such man-in-the-middle attacks.¹⁰ After considerable research into understanding and improving the speed at which software is patched,^{14,22} much of software patching has become automated. However, the web's PKI requires a surprising amount of *manual administration*. To revoke a certificate, website administrators must send a request to their Certificate Authority (CA), and this request may be manually reviewed before the certificates are finally added to a list that browsers (are supposed to) check. Such operations occur at human timescales (hours or days) instead of computer ones (seconds or minutes). An important open question is: when private keys are compromised, how long are SSL clients exposed to potential attacks?

Historically, these manual processes have been difficult to measure: how can one measure, at scale, how long these processes take if we do not know how often, or precisely when, administrators realize their keys are compromised? In this paper, we use a widespread security vulnerability from 2014, Heartbleed, as a natural experiment: the moment Heartbleed was announced, all administrators of vulnerable servers should have initiated their manual processes as quickly as possible.³ This natural experiment allows us to measure at scale the manual administration of the web's PKI. In particular, this paper focuses on the response to the public announcement of Heartbleed, in terms of how quickly certificates were reissued and whether or not the certificates were eventually revoked.

Our results expose incomplete and slow administrative practices that ultimately weaken the security of today's PKI. On the positive side, we also identify ways in which the PKI can be strengthened. Our hope is that, through better understanding how the PKI operates in practice, the security and research community can take concrete steps toward improving this system on which virtually all Internet users rely.

2. BACKGROUND

In this section, we review the relevant background of SSL/TLS and the PKI, and we describe the Heartbleed vulnerability that serves as our natural experiment.

The original version of this paper was published in the *2014 ACM Internet Measurement Conference (IMC'14)*.

2.1. Certificates

One of the critical components of the PKI is a *certificate*: a signed attestation binding a human-understandable *subject* (a domain name or business name) to a public key. Certificates are signed by a CA, who in turn has its own certificate, etc., forming a logical chain that terminates at a self-signed *root certificate*. By issuing a certificate, a CA is essentially asserting “this subject is the sole owner of the private key corresponding to this certificate’s public key.” Thus, if someone can prove knowledge of the private key (e.g., by using it to sign a message), then this proves that this someone is the subject. As a result, *anyone* who has knowledge of the private key can pretend to be that subject.

The assumption that only legitimate subjects hold the corresponding private keys is central to the PKI’s ability to authenticate servers. Unfortunately, keys can be compromised. For example, software vulnerabilities in SSL implementations have resulted in predictable keys²² or the ability to read sensitive server-side data.⁶

When a private key is compromised, a responsible administrator must do at least three things: first, the administrator must *patch* the vulnerable software. Second, because the certificate’s private key has been compromised, the administrator must generate a new key pair and ask their CA to *reissue* a new certificate with this new private key. However, the old, compromised certificate would still exist, and it could be used by a malicious party to undetectably impersonate the website. Thus, there is a critical third step an administrator must do in response to a key compromise: *revoke* the old certificate. It is important to note that the final two steps necessarily involve the CA, and many CAs charge for these actions, which may lead to perverse incentives for site owners.

2.2. Certificate reissue

When a website stops using a certificate—for instance, because the certificate has been compromised, or because it expired—they must obtain a new certificate. This process is referred to as *reissuing* the certificate. To do so, the system administrator must contact the CA who signed their certificate and request a new certificate and signature. In cases where the private key may have been compromised, the administrator should also choose a new public/private key pair (since reissuing the certificate with the old public key does nothing to mitigate attacks that leverage the leaked private key).

2.3. Certificate revocation

A *certificate revocation* is another signed attestation from a CA, which essentially states “this certificate should no longer be considered valid.” CAs are responsible for making revocations available for download, and typically do so with Certificate Revocation Lists (CRLs). Browsers (are supposed to) download CRLs to check if a presented certificate has been revoked; the longer an administrator waits to revoke compromised certificates, the longer users are susceptible to man-in-the-middle attacks.

The PKI uses a default-valid model, where potentially compromised certificates remain valid until their expiration

date or until they are revoked. The security of any PKI is thus critically dependent on the *timeliness* of certificate revocations. However, requesting a revocation is a surprisingly *manual* process, typically requiring an administrator to visit a website, fill in a form, provide a reason for the revocation, and wait for a representative at the CA to manually inspect the request before issuing the revocation.

While it seems natural to assume that certificates are reissued at precisely the moment the old certificate is revoked, in fact today’s PKI protocols make no such requirement. As our study will demonstrate, reissues can happen before, during, or after a revocation—or even without revoking the old certificate at all. To the best of our knowledge, we are the first to correlate revocations with reissues.

2.4. Heartbleed

Heartbleed is a buffer over-read vulnerability discovered in OpenSSL¹⁵ versions 1.0.1 (released March 14, 2012) through 1.0.1f. The vulnerability stems from a bug in OpenSSL’s implementation of the TLS Heartbeat Extension.¹⁷ The intended functionality of TLS Heartbeat is to allow a client to test a secure communication channel by sending a “heartbeat” message consisting of a string and the 16-bit *payload_length* of this string. Unfortunately, vulnerable OpenSSL versions fail to check that the *payload_length* supplied by the client matches the length of the provided string. This allows a malicious client to craft a heartbeat message containing a 1-byte string and $2^{16} - 1$ as the *payload_length*. In this case, OpenSSL will allocate a 64KB block of heap memory, *memcpy()* 64KB of data into it, starting with the 1-byte string, and finally send the contents of the entire buffer to the client. This allows the malicious client to read up to $2^{16} - 2$ bytes of the server’s heap memory, although the client cannot choose *which* memory is read.

By repeatedly exploiting Heartbleed, an attacker can extract sensitive data from the server, including SSL private keys.¹⁹ To make matters worse, OpenSSL does not log heartbeat messages, giving attackers free reign to undetectably exploit Heartbleed. Given the severity and undetectable nature of Heartbleed, site operators were urged to immediately update their OpenSSL software and revoke and reissue their certificates.³

Why study Heartbleed? Heartbleed was first discovered by Neel Mehta from Google on March 21, 2014. On April 7, the bug became public and the OpenSSL project released a patched version (1.0.1g) of the OpenSSL library.⁸

The significance of this timeline, and of Heartbleed in general, is that it represents a point in time after which the administrators of *all* vulnerable servers *should have* (1) patched their server, (2) revoked their old certificate, and (3) issued a new one. The scope of this vulnerability—it is estimated that up to 17% of all HTTPS web servers were vulnerable¹³—makes it an ideal case study for evaluating large-scale properties of SSL security in the face of private key compromise. As a result, Heartbleed acts as a *natural experiment*, allowing us to measure how completely and quickly administrators took steps to secure their keys. While such events are (sadly) not uncommon,²² the intense press

coverage surrounding Heartbleed reduces the likelihood that administrators failed to take action because they were unaware of the vulnerability.

3. DATA AND METHODS

We now describe the data sets that we collected and our methodology for determining a host's SSL certificate, when it was in use, if and when the certificate was revoked, and if the host was (or is still) vulnerable to the Heartbleed bug.

3.1. Certificate data source

We obtain our collection of SSL certificates from (roughly) weekly scans of the entire IPv4 address space made available by Rapid7.¹⁶ We use scans collected between October 30, 2013 and April 28, 2014. There are a total of 28 scans during this period, giving an average of 6.7 days (with a minimum of 3 days and maximum of 9 days) between successive scans.

The scans found an average of 26.9 million hosts responding to SSL handshakes on port 443 (an average of 9.12% of the entire IPv4 address space). Across all of the scans, we observed a total of 19,438,865 unique certificates (including all leaf and CA certificates). In the sections below, we describe how we filtered and validated this data set; an overview of the process is provided in Figure 1.

3.2. Filtering data

To focus on web destinations that are commonly accessed by users, we use the Alexa Top-1M domains¹ as observed on April 28, 2014. We first extract all leaf (non-CA) certificates that advertise a *Common Name* (CN) that is in one of the domains in the Alexa list (e.g., we would include certificates for facebook.com, www.facebook.com, as well as *.dev.facebook.com). This set represents 1,573,332 certificates (8.1% of all certificates).

Unfortunately, despite leaf certificates having a CN in the Alexa list, many may not be valid (e.g., expired certificates, forged certificates, certificates signed by an unrecognized root, etc.). We removed these invalid certificates⁴ by running `openssl verify` on each certificate (and its corresponding chain). We configure OpenSSL to trust the root CA certificates included by default in the OS X 10.9.2 root store¹²; this includes 222 unique root certificates.

After validation, we are left with 628,692 leaf certificates (40.0% of all certificates advertising Alexa domains and 3.2% of all certificates). We refer to this set of certificates as the *Leaf Set*, each of which has a valid chain. We refer to the set of all CA certificates on these chains (not including the leaf certificates) as the *CA Set*, which contains 910 unique certificates. The Leaf Set certificates cover 166,124 (16.6%) of the Alexa Top-1M domains. This is the set of certificates and chains that we use in the remainder of the paper.

3.3. Collecting CRLs

To determine if and when certificates were revoked, we extracted the CRL URLs out of all Leaf Set certificates. We found 626,659 (99.7%) of these certificates to include at least one well-formed, reachable CRL URL. For certificates that included multiple CRL URLs, we included them all. We found a total of 1,386 unique CRL URLs (most certificates use a unified CRL provided by the signing CA, so the small number of CRLs is not surprising). We downloaded all of these CRLs on May 6, 2014, and found 45,268 (7.2%) of the Leaf Set certificates to be revoked.

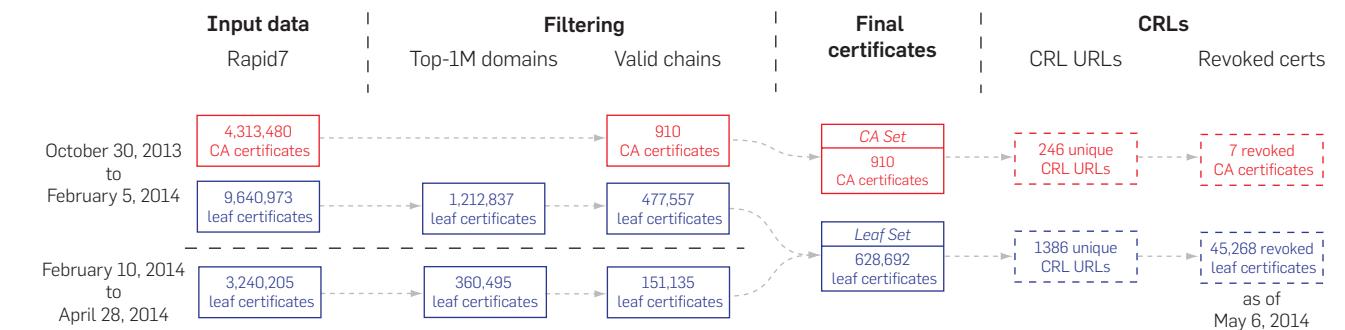
We also collected the CRL URLs for all certificates in the CA set. We found that 884 (97.1%) of the certificates in the CA Set included a reachable CRL; the union of these URLs comprised 246 unique reachable URLs. We downloaded these CRLs on May 6, 2014, as well. We found a total of seven CA certificates that were revoked, which invalidated 60 certificates in the Leaf Set (< 0.01%).

3.4. Inferring the Heartbleed vulnerability

Finally, we wish to determine if a site was ever vulnerable to the Heartbleed OpenSSL vulnerability (and if it continued to be vulnerable at the end of the study). Doing so allows us to reason about whether the site operators should have reissued their SSL certificate(s) and revoked their old one(s). Determining if a host is currently vulnerable to Heartbleed is relatively easy, as one can simply send an improperly-formatted SSL heartbeat message with a `payload_length` of 0 to test for vulnerability without exfiltrating any data.⁶

However, determining if a site *was* vulnerable in the past—but has since updated their OpenSSL code—is more challenging. We observe that only three of the common TLS

Figure 1. Workflow from raw scans of the IPv4 address space to valid certificates (and corresponding CRLs) from the Alexa Top-1M domains. The Rapid7 data after February 5, 2014 did not include the intermediate (CA) certificates, necessitating additional steps and data to perform validation.



implementations have ever supported SSL Heartbeats¹⁷: OpenSSL,¹⁵ GnuTLS,²⁰ and Botan.² Thus, if a host supports the SSL Heartbeat extension, we know that it is running one of these three implementations. Botan is targeted for client-side TLS, and we know of no popular web server that uses the Botan TLS library. GnuTLS has support for the SSL Heartbeat extension, but it is not enabled by default. Furthermore, GnuTLS supports the Max Fragment Length SSL extension,⁷ which is enabled by default, while OpenSSL has never supported this extension. Thus, if we observe a host that supports SSL Heartbeat but *not* the Max Fragment Length, we declare that host to have been running a vulnerable version of OpenSSL.

To collect the list of sites that were ever vulnerable to Heartbleed, we extracted the IP addresses in the April 28, 2014 Rapid7 scan that were advertising a certificate with a CN in the Alexa Top-1M list. We found 5,951,763 unique IP addresses in this set. We then connected to these IP addresses on port 443, determined the SSL extensions that the host supported, and checked whether the host was still vulnerable to the Heartbleed vulnerability.

4. ANALYSIS

We now examine the collected SSL certificate data, beginning with a few definitions.

4.1. Definitions

We are concerned with the evolution of SSL certificates (i.e., when are new certificates created, old ones retired, etc.). To aid in understanding this evolution, we define the following notions:

Certificate birth: We define the birth of an SSL certificate to be the date of the first scan where we observed any host advertising that certificate.

Certificate death: Defining the death of a certificate is more complicated, as we observe a number of instances where many hosts advertise a given certificate, and then all but a few of the hosts switch over to a new certificate (presumably, the site intended to retire the old certificate, but failed to update some of the hosts). To handle these cases, we calculate the *maximum* number of hosts that were ever advertising each certificate. We then define the death of an SSL certificate to be the last date that the number of hosts advertising the certificate was above 10% of that certificate's maximum. The 10% threshold prevents us from incorrectly classifying certificates that are still widely available as dead, even if the certificate has been reissued.

An example of certificate lifetime for m.scotrail.co.uk is shown in Figure 2. All hosts except one switch to a new certificate after February 10, 2014; this lone host finally switches on April 28, 2014. In this case, we would consider the death date of the old certificate to be February 10, 2014.

Based on these definitions, we can now define the notion of a certificate reissue and revocation:

Certificate reissue: We consider a certificate to be reissued if the following three conditions hold: (a) we observe the certificate die, and (b) we observe a new certificate for the same *Common Name* born within 10 days of the certificate's death,

and (c) we observe at least one IP address switch from the old certificate to the new. We define the date of the certificate reissue to be the date of the certificate's death. For the sake of clarity, we refer to the old certificate that was replaced as the *retired certificate*.

Certificate revocation: A certificate is revoked if its serial number appears in any of the certificate's CRLs. The date of revocation is provided in the CRL entry.

In Figure 3, we present the number of certificate births, deaths, reissues, and revocations per day over time (please note the y-axis is in log scale). Births almost always outnumber deaths, meaning that the total number of certificates in-the-wild is growing. Furthermore, we see an average of 29 certificate revocations per day before Heartbleed; after Heartbleed, this jumps to an average of 1,414 revocations per day.

4.2. Server patching

We present a brief analysis of the number of certificates hosted by machines that were ever vulnerable to Heartbleed. Of the 428,552 leaf certificates that were still alive on the last scan, we observe 122,832 (28.6%) of them advertised by a host that was likely vulnerable to Heartbleed at some point in time. These certificates come from 70,875 unique Alexa Top-1M domains. Of these certificates, 11,915 (from 10,366 unique domains) were on hosts that were *still* vulnerable at the time of our crawl (April 30, 2014). This result demonstrates that even in the wake of a well-publicized, severe security vulnerability, around 10% of vulnerable sites did not address the issue three weeks after the fact.

Figure 2. Example of lifetime, for certificates for m.scotrail.co.uk. All hosts except one switch to a new certificate after February 10, 2014.

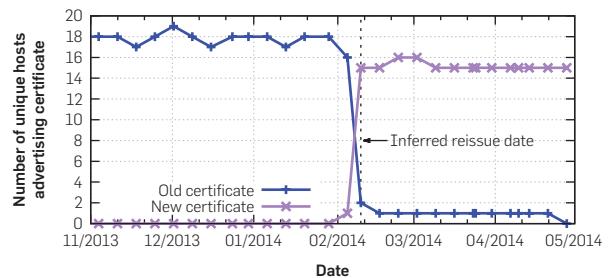
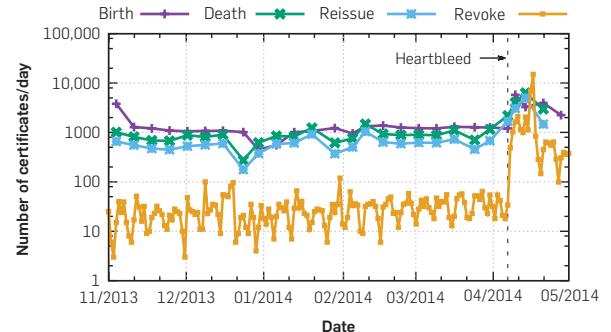


Figure 3. The rate of certificate births, deaths, reissues, and revocations increased sharply after the Heartbleed announcement.



In Figure 4, we present the fraction of domains that have at least one SSL host that was ever vulnerable to Heartbleed (or still was as of April 30, 2014). We can observe a slight increase in likelihood of ever being vulnerable for the most popular sites, but the distribution quickly stabilizes. The increased likelihood of being vulnerable is likely because these sites have larger numbers of hosts. This trend is mirrored in the hosts that are still vulnerable on April 30, 2014.

4.3. Certificate reissues

We now examine the reissuing of SSL certificates in the wake of Heartbleed. Not all SSL certificate reissues that we observe following Heartbleed's announcement are due to the Heartbleed vulnerability. In particular, reissues can happen for at least two other reasons: first, the old certificate could be expiring soon. For example, before Heartbleed, we observe that 50% of certificates are reissued within 60 days of their expiry date. Second, a site may periodically reissue certificates as a matter of policy (regardless of expiration date). For example, we observed that Google typically reissues the google.com certificate every two weeks.

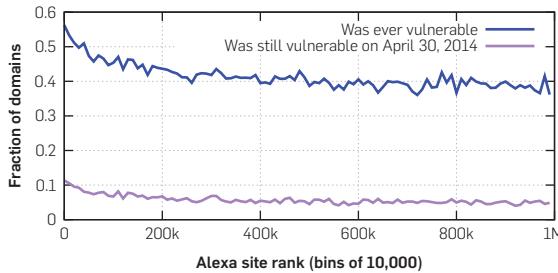
In this study, we wish to distinguish a *Heartbleed-induced* certificate reissue from a reissue that would have happened anyway. We define a certificate reissue to be Heartbleed-induced if all three of the following conditions hold:

1. The date of reissue was on or after April 7, 2014 (the day Heartbleed was announced).
2. The certificate that is reissued was not going to expire for at least 60 days. This eliminates certificates that were likely to be reissued in the near future anyway.
3. We do not observe more than two other reissues for certificates with that CN in the time before Heartbleed. This implies that certificates with that name do not typically get reissued more than once every three months.

Thus, for the examples discussed so far, we do not consider the reissue of the retired certificate in Figure 2 to be Heartbleed-induced (as it happened before Heartbleed), and we do not consider any of the google.com reissues to be Heartbleed-induced (because we observed a total of 12 reissues of that certificate prior to Heartbleed).

Heartbleed-induced reissues. Overall, we observe 36,781 certificate reissues that we declare to be Heartbleed-induced

Figure 4. Fraction of domains that have at least one host that was ever vulnerable to Heartbleed as a function of Alexa rank, as well as domains that continued to be vulnerable at the end of the study.



in the three weeks following the announcement; this is 8.9% of all certificates that were alive at the time Heartbleed was announced. Figure 5 examines the fraction of sites that have at least one Heartbleed-induced certificate reissue, as a function of Alexa rank; we can observe a strong correlation with Alexa rank. Higher-ranked sites are much more likely to have reissued at least one certificate due to Heartbleed (even though they are only slightly more likely to have been vulnerable, as observed in Figure 4). This result complements previous studies' findings that more popular websites often exhibit more sound administrative practices.^{5,9}

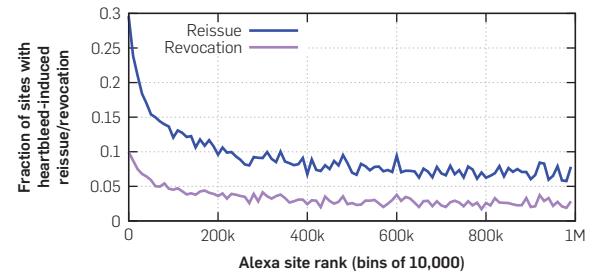
Reissues with same key. System administrators who believe that their SSL private key may have been compromised should generate a new public/private key pair when reissuing their certificate. We now examine how frequently this is done, both in the case of normal certificate reissues and for Heartbleed-induced reissues.

Before Heartbleed, we observe that reissuing a certificate using the same key pair is quite common; up to 53% of all reissued certificates do so. This high-level of key reuse is at least partially due to system administrators re-using the same Certificate Signing Request (CSR) when requesting the new certificate from their CA. After Heartbleed, we observe a significant drop in the frequency of reissuing certificates with the same key, that is, sites are generating a new key pair more frequently. However, if we focus on the Heartbleed-induced reissues, we observe that a non-trivial fraction (4.1%) of these certificates are reissued with the same key (thereby defeating the purpose of reissuing the certificate). In fact, we observe a total of 912 such certificates coming from 747 distinct Alexa domains; these certificates may represent cases where administrators believe they have correctly responded to Heartbleed, but their certificates remain as vulnerable as if they had not reissued at all.

Vulnerable certificates. Finally, we examine the certificates that *should have* been reissued (regardless of whether they actually were); we refer to these certificates as *vulnerable certificates*. We declare a certificate to be vulnerable if the following three conditions hold:

1. Its date of birth was before April 7, 2014.
2. It has not expired as of April 30.
3. It was advertised by at least one host that was (or is) vulnerable to Heartbleed.

Figure 5. Fraction of domains that have at least one Heartbleed-induced reissue/revocation as a function of Alexa rank.



In other words, these certificates are vulnerable because their private keys could have been stolen by attackers.

Overall, we find 107,712 vulnerable certificates. Of these, only 28,652 (26.7%) have been reissued as of April 30. The remaining 79,060 (73.3%) vulnerable certificates that have not been reissued come from 55,086 different Alexa Top-1M domains. Thus, the vast majority of SSL certificates that were potentially exposed by the Heartbleed bug remain in-use over three weeks after the vulnerability was announced.

4.4. Certificate revocation

We now turn to investigating certificate revocation before, during, and after the revelation of Heartbleed. Recall that it is critical that a vulnerable certificate be revoked: even if a site reissues a new certificate, if an attacker gained access to the vulnerable certificate's private key, then that attacker will be able to impersonate the owner until either the certificate expires or is revoked. We study both revocation and expiration here, and correlate them with rates of reissue.

Overall revocation rates. Figure 3 shows the number of certificate revocations over time. As noted above, the average jumps from 29 revocations per day to 1,414 post-Heartbleed. However, the spike on April 16, 2014 is somewhat misleading, as it was largely due to the mass-revocation of 19,384 CloudFlare certificates.¹⁸

To mitigate this issue, we plot in Figure 6 the number of unique *domains* that revoked at least one certificate over time. We make three interesting observations: *First*, the magnitude of the Heartbleed-induced spike is greatly reduced, but we still observe an up-to-40-fold increase in the number of domains issuing revocations per day. *Second*, we observe that the number of domains issuing revocations falls closer to its pre-Heartbleed level by April 28, suggesting that within 3 weeks *most of the domains that will revoke their certificate in direct response to Heartbleed already have*.

Third, we observe three “dips” in the post-Heartbleed revocation rate on April 13, April 20, and April 27—all weekends, indicating that far fewer revocations occur on the weekend relative to the rest of the week. This periodicity can also be (less-easily) observed in the pre-Heartbleed time frame. It is reasonable to assume revocations dip on weekends because humans are involved in the revocation process, however it is not clear who is responsible for the delays: is it site administrators or CRL maintainers at CAs (or both) who are not working on weekends? Regardless of

who is responsible, these weekend delays are problematic for online security, since vulnerabilities (and the attackers who exploit them) do not take weekends off.

Heartbleed-induced revocations. Similar to certificate reissues, not all certificate revocations after April 7, 2014 are necessarily due to Heartbleed (e.g., the site could have exposed their private key due to a different vulnerability). We therefore define a *Heartbleed-induced revocation* to be a certificate revocation where the certificate had a Heartbleed-induced reissue (see Section 4.3).

Overall, we observe 14,726 Heartbleed-induced revocations; this corresponds to 40% of all Heartbleed-induced reissued certificates. Thus, 60% of all certificates that were reissued due to Heartbleed were *not* revoked, implying that, if the certificate's private key was actually stolen, the attacker still would be able to impersonate the victim without any clients being able to detect it.

Figure 5 presents the fraction of sites that have at least one Heartbleed-induced certificate revocation, as a function of Alexa rank. Similar to reissues, sites with high rank are slightly more likely to revoke. Ideally, the two lines in Figure 5 should be coincident, that is, all sites reissuing certificates due to Heartbleed should also have revoked the retired certificates. This result highlights a serious gap in security best-practices across all of the sites in the Alexa Top-1M.

Finally, we examine *revocation delay*, or the number of days between when a certificate is reissued and it is revoked. Figure 7 presents the cumulative distribution of the revocation delay for both Heartbleed-induced and non-Heartbleed-induced revocations. To make the distributions comparable, we only look at differences between -10 and 10 days (recall that Heartbleed-induced reissues and revocations can only occur after April 7, 2014, limiting that distribution). We observe that Heartbleed-induced revocations appear to happen slightly more quickly, though not to the extent one might expect, given the urgent nature of the vulnerability. We also observe that revocation almost always happens *after* reissue, which makes sense, since this preserves the availability of HTTPS websites. This result contradicts previous assumptions⁵ that revocations and reissues occur simultaneously.

Expirations are not enough. To demonstrate how long the effects of the Heartbleed vulnerability will be felt if sites do not revoke their vulnerable certificates, we analyze vulnerable certificates that, by the end of our data collection, were

Figure 6. The rate of domains revoking certificates spiked after Heartbleed, but dropped closer to normal after three weeks.

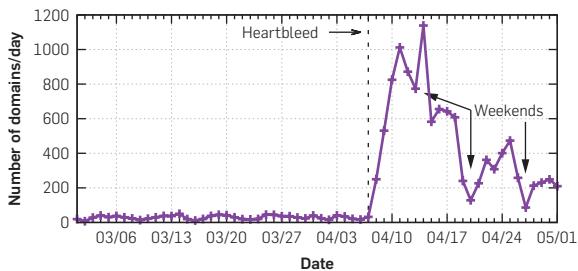
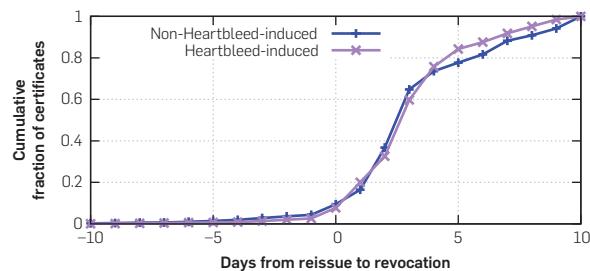


Figure 7. Heartbleed-induced revocations were issued slightly faster than other revocations.



reissued but never revoked. Although we find that 60% of the certificates expire within a year, there are vulnerable certificates that are valid for up to 5 years after Heartbleed was announced. In fact, 10% of the vulnerable certificates still had over 3 years of validity remaining. We conclude from this that, given the meager rates of revocation, it would be helpful for CAs to shift to shorter expiry times in their certificates.

Reissues and revocation speed. Next, we examine how quickly sites responded to Heartbleed. Figure 8 shows the fraction of vulnerable certificates that were not reissued or revoked over the three weeks following the Heartbleed announcement. In this figure, the initial y values do not all start at 1.0 for reissues: this is because, with the coarse granularity of our data, we know the range of time during which some certificates were reissued, but not the precise day. We therefore provide the *most optimistic* possibility: if we know a certificate was reissued between days d and $d + k$, we assume it was reissued on day d .

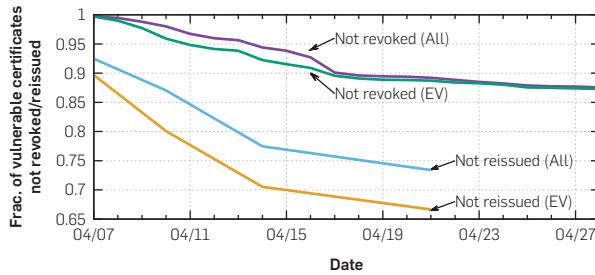
This figure presents a bleak view of how *thoroughly* sites revoke and reissue their certificates (note that the y-axis begins at 0.60). Three weeks after the revelation of Heartbleed, over 87% of all certificates we found to be vulnerable were not revoked, and over 73% of them were not reissued. We also found that the revocation rate follows a pattern previously observed in earlier studies on the spread of patches^{14, 22}: there is an exponential drop-off, followed by a gradual decline. This behavior is even more pronounced when looking farther beyond the Heartbleed announcement: 16 weeks after the announcement, there were still 86% who had not revoked and 70% who had not reissued.

Extended validation certificates. Recall that one of the major roles of a CA is to validate the identity of the subjects who purchase certificates. *Extended Validation* (EV) certificates are a means by which CAs can express that this identity-verification process has followed (presumably) more stringent criteria. Many browsers present EV certificates differently in the address bar.

EV certificates are standard X.509 certificates that are not, in and of themselves, more secure, but the rationale is that with a more thorough verification process by the CAs, these certificates deserve greater trust. That said, there remains concern as to whether this trust is well-placed. We close by investigating the rate at which vulnerable EV certificates were revoked and reissued as compared all certificates overall.

Overall, Figure 8 shows EV certificates follow similar trends to the entire corpus, with a slightly faster and more

Figure 8. Many vulnerable certificates were not revoked and reissued after Heartbleed (note that the y-axis does not begin at zero).



thorough response. Interestingly, while EV certificates were revoked more quickly, their non-EV counterparts caught up within 10 days; however, EV certificates were reissued both more quickly and more thoroughly. We expect that the underlying cause of this observation is a self-selection effect, that is, security-conscious sites are more likely to seek out EV certificates in the first place. Nonetheless, there are still many vulnerable EV certificates that have not been reissued three weeks after the event (67%) and that have not been revoked three weeks after (87%).

5. CONCLUDING DISCUSSION

In this paper, we studied how SSL certificates are reissued and revoked in response to a widespread vulnerability, Heartbleed, that enabled undetectable key compromise. We conducted large-scale measurements and developed new methodologies to determine how the most popular one million domains reacted to this vulnerability in terms of certificate management, and how this impacts security for clients.

We found that the vast majority of vulnerable certificates have not been reissued. Further, of those domains that reissued certificates in response to Heartbleed, 60% did not revoke their vulnerable certificates—if they do not eventually become revoked, 20% of those certificates will remain valid for two or more years. The ramifications of these findings are alarming: Web browsers will remain potentially vulnerable to malicious third parties using stolen keys for a long time to come. Additionally, we found that domains with EV certificates performed only marginally better than other domains with respect to reissuing and revocation.

Our results are, in some ways, in line with previous studies on the rates at which administrators patched vulnerable software²²—for instance, revocation rates followed a sharp exponential drop-off shortly after the vulnerability was made public, and tapered off soon thereafter. However, unlike software bugs, we find that the vast majority of certificates remain vulnerable to attacks, as they have still not been reissued or revoked. These findings indicate that the current practices of certificate management are misaligned with what is necessary to secure the PKI.

5.1. Surveying system administrators

To help better understand the reasons behind the lack of prompt certificate reissues and revocations, we informally surveyed a few systems administrators. We asked what steps they had taken in response to Heartbleed: did they patch, reissue, and revoke, and if not, then why not? We received seven responses. Most reported manually patching their systems, but some relied on managed servers or automatic updates and therefore took no Heartbleed-specific steps. There was some variance in when patches were applied, due to a combination of scheduled reboots and delayed responses from vendors, but the majority of patches were applied quickly.

For revoking and reissuing, however, we saw a wide spectrum of behavior. The few who reissued and revoked did so within 48 hours. Many neither revoked nor reissued; a common reason provided was that the vulnerable hosts were not hosting sensitive data or services. Along similar

lines, others reported having reissued the certificate but not revoking, explaining that the certificate is only for internal use. Finally, others reported that they did not perceive reissuing and revoking as important because they had patched quickly after the bug was publicly announced (recall, however, that the vulnerability was introduced over 2 years prior).

Our results from this small survey should be viewed anecdotally—more extensive surveys on certificate administration would be an important area of future work⁶—but they do shed light on the root causes of why revoking and reissuing are not on equal footing with patching. While administrators almost universally understand the importance of patching after a vulnerability, many do not appreciate or know about the importance of revoking and reissuing certificates with new keys. Of those administrators who do understand the importance, some reported push-back from others who perceived the process as being overly complex. In sum, this points to the need for broader education on the treatment of certificates, and perhaps more assistance from CAs to help ensure that all the prescribed steps are taken.

5.2. Lessons learned

Our results suggest several changes to common PKI practices that may improve security. First, low revocation rates and long expiration dates form a dangerous combination. Techniques that automate revocation would vastly reduce the period during which clients are vulnerable to malicious third parties. Similarly, adopting short certificate expiration dates (as suggested by Topalovic et al.²¹) by default will significantly reduce the validity period of vulnerable certificates. Second, mechanisms that enable simultaneous reissue-and-revoke for certificates will make it less likely that invalid certificates are accepted by clients. *Third*, we have found that many domains continue to serve old, vulnerable certificates even after they reissue. Given the large number of certificates and hosts using them per domain in our dataset, we believe administrators would benefit from tools that more easily track and validate the set of certificates they are using.

5.3. Future work

This paper is, we believe, the first step towards understanding the manual process of reissuing and revoking certificates in the wake of a vulnerability. Several interesting open problems remain. Because our data focuses on the server and CA side of the PKI ecosystem, we are unable to draw any direct conclusions as to what clients experience. A host-centered measurement study would allow us to understand not only when revocations were added to CRLs, but when clients actually received the CRLs. Moreover, our study opens many questions as to *why* the certificate reissue and revocation processes are so extensively mismanaged. Our results reinforce previous findings that site popularity is correlated with good security practices, but even the highest ranked Alexa websites show relatively anemic rates of reissues and revocations. Understanding the root causes for why the PKI is mismanaged is an important step toward developing a secure infrastructure.

5.4. Open source

Our analysis relied on both existing, public sources of data and those we collected ourselves. We make all of our data and our analysis code available to the research community at <https://securepki.org>. C

References

1. Alexa Top 1 Million Domains. <http://s3.amazonaws.com/alexa-static/top-1m.csv.zip>.
2. Botan SSL Library. <http://botan.randombit.net>.
3. CERT Vulnerability Note VU#720951: OpenSSL TLS heartbeat extension read overflow discloses sensitive information. <http://www.kb.cert.org/vuls/id/720951>.
4. Chung, T., Liu, Y., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Wilson, C. Measuring and applying invalid SSL certificates: The silent majority. In *ACM Internet Measurement Conference (IMC)* (2016).
5. Durumeric, Z., Kasten, J., Bailey, M., Halderman, J.A. Analysis of the HTTPS certificate ecosystem. In *ACM Internet Measurement Conference (IMC)* (2013).
6. Durumeric, Z., Kasten, J., Li, F., Amann, J., Beekman, J., Payer, M., Weaver, N., Halderman, J.A., Paxson, V., Bailey, M. The matter of Heartbleed. In *ACM Internet Measurement Conference (IMC)* (2014).
7. Eastlake, D III. Transport Layer Security (TLS) Extensions: Extension Definitions, Jan. 2011. IETF RFC-6066.
8. Grubbs, B. Heartbleed disclosure timeline: who knew what and when, 2014. <http://www.smh.com.au/it-pro/security-it/heartbleed-disclosure-timeline-who-knew-what-and-when-20140415-zqurk.html>.
9. Holz, R., Braun, L., Kammenhuber, N., Carle, G. The SSL Landscape – A thorough analysis of the X.509 PKI using active and passive measurements. In *ACM Internet Measurement Conference (IMC)* (2011).
10. Huang, L.-S., Rice, A., Ellingsen, E., Jackson, C. Analyzing forged SSL certificates in the wild. In *IEEE Symposium on Security and Privacy (S&P)* (2014).
11. Liu, Y., Tome, W., Zhang, L., Choffnes, D., Levin, D., Maggs, B.M., Mislove, A., Schulman, A., Wilson, C. An end-to-end measurement of certificate revocation in the web's PKI. In *ACM Internet Measurement Conference (IMC)* (2015).
12. Mac OS X 10.9.2 Root Certificates. <http://support.apple.com/kb/HT6005>.
13. Mutton, P. Half a million widely trusted websites vulnerable to heartbleed bug, 2014. <http://news.netcraft.com/archives/2014/04/08/half-a-million-widely-trusted-websites-vulnerable-to-heartbleed-bug.html>.
14. Nappa, A., Johnson, R., Bilge, L., Caballero, J., Dumitraș, T. The attack of the clones: A study of the impact of shared code on vulnerability patching. In *IEEE Symposium on Security and Privacy (S&P)* (2015).
15. OpenSSL Project. <https://www.openssl.org>.
16. Rapid7 SSL Certificate Scans. <https://scans.io/study/sonar.ssl>.
17. Seggelmann, R., Tuexen, M., Williams, M. Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension, Feb. 2012. IETF RFC-6520.
18. Sullivan, N. The Heartbleed Aftermath: all CloudFlare certificates revoked and reissued, 2014. <http://blog.cloudflare.com/the-heartbleed-aftermath-all-cloudflare-certificates-revoked-and-reissued>.
19. Sullivan, N. The Results of the CloudFlare Challenge, 2014. <http://blog.cloudflare.com/the-results-of-the-cloudflare-challenge>.
20. The GnuTLS Transport Layer Security Library. <http://www.gnutls.org>.
21. Topalovic, E., Saeta, B., Huang, L.-S., Jackson, C., Boneh, D. Toward short-lived certificates. In *Web 2.0 Security & Privacy (W2SP)* (2012).
22. Yilek, S., Rescorla, E., Shacham, H., Enright, B., Savage, S. When private keys are public: Results from the 2008 Debian OpenSSL vulnerability. In *ACM Internet Measurement Conference (IMC)* (2009).

Liang Zhang, David Choffnes, Alan Mislove, and Christo Wilson ([liang, choffnes, amislove, cbw]@ccs.neu.edu), Northeastern University, Boston, MA, USA.

Dave Levin and Tudor Dumitraș ([dml@cs, tdumitra@umiacs].umd.edu), University of Maryland, College Park, MD, USA.

Aaron Schulman (aschulm@stanford.edu), Stanford University, Stanford, CA, USA.

CAREERS

Harvard John A. Paulson School of Engineering and Applied Sciences Tenured Professor in Computer Science

The Harvard John A. Paulson School of Engineering and Applied Sciences (SEAS) seeks applicants for a position at the tenured level in the area of Artificial Intelligence with Societal Impact, with an expected start date of July 1, 2018.

We seek a computer scientist whose research accomplishments include fundamental advances in AI and impact through applications that improve societal well-being. We seek candidates who have a strong research record and a commitment to undergraduate and graduate teaching and training. We particularly encourage applications from historically underrepresented groups, including women and minorities.

Computer Science at Harvard is enjoying a period of substantial growth in numbers of students and faculty hiring, and in expanded facilities. We benefit from outstanding undergraduate and graduate students, world-leading faculty, an excellent location, significant industrial collaboration, and substantial support from the Harvard Paulson School. For more information, see <http://www.seas.harvard.edu/computer-science>.

The associated Center for Research on Computation and Society (<http://crcs.seas.harvard.edu/>), Berkman Klein Center for Internet & Society (<http://cyber.harvard.edu>), Data Science Initiative (<https://datascience.harvard.edu/>), and Institute for Applied Computational Science (<http://iacs.seas.harvard.edu>) foster connections among computer science and other disciplines throughout the university.

Candidates are required to have a doctoral degree in computer science or a related area.

Required application documents include a cover letter, CV, a statement of research interests, a teaching statement, and up to three representative papers. Candidates are also required to submit the names and contact information for at least three references. Applicants can apply online at <https://academicpositions.harvard.edu/postings/8037>.

We are an equal opportunity employer and all qualified applicants will receive consideration for employment without regard to race, color, religion, sex, sexual orientation, gender identity, national origin, disability status, protected veteran status, or any other characteristic protected by law.

IU School of Informatics and Computing, IUPUI Associate Dean for Research

The Indiana University School of Informatics and Computing (SoIC), IUPUI campus, invites applications for a tenured associate or full professor in the growing field of data science (or related area) to fill the position of Associate Dean for Research. The appointment will begin August 1, 2018. Can-

didates must demonstrate an outstanding scholarly record of research, exhibited by high-impact peer-reviewed publications, a forward-looking, vigorous research agenda and a demonstrated history of securing significant, competitive external funding.

An exceptional researcher is sought to lead and expand the research enterprise of our school and contribute to the department's growing data science academic program. All areas of data science will be considered including data mining, statistical machine learning, descriptive, predictive, and prescriptive analytics, cloud computing, distributed databases, high performance computing, data visualization, or other areas involving the collection, organization, management, and extraction of knowledge from massive, complex, heterogeneous datasets. Data may include text, images, video, sensor and instrument data, clickstream data, social media interactions, neuroimaging data, genomics, proteomics, or metabolomics data, etc.

The overarching responsibility is to expand the research portfolio of the SoIC. Full details of this position can be found at <https://indiana.peopleadmin.com/postings/5287>

Questions pertaining to this position can be directed to Jeff Hostetler, Assistant to the Dean at jehostet@iupui.edu

The School of Informatics and Computing is eager to consider applications from women and minorities. Indiana University is an Affirmative Action/Equal Opportunity Employer. IUPUI is an Affirmative Action/Equal Opportunity Institution M/F/D/V.

National University of Singapore (NUS) Sung Kah Kay Assistant Professorship

The Department of Computer Science at the National University of Singapore (NUS) invites applications for the Sung Kah Kay Assistant Professorship. Applicants can be in any area of computer science. This prestigious chair was set up in memory of the late Assistant Professor Sung Kah Kay after his untimely demise early in his career at NUS. Candidates should be early in their academic careers and yet demonstrate outstanding research potential, and strong commitment to teaching.

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 boasts 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.

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 the position is filled

If you have further enquiries, please contact the Search Committee Chair, Weng-Fai Wong, at csrec@comp.nus.edu.sg.

Southern University of Science and Technology (SUSTech)

Professor Position in Computer Science and Engineering

The Department of Computer Science and Engineering (CSE, <http://cse.sustech.edu.cn/en/>), Southern University of Science and Technology (SUSTech) has multiple Tenure-track faculty openings at all ranks, including Professor/Associate Professor/Assistant Professor. We are looking for outstanding candidates with demonstrated research achievements and keen interest in teaching.

Applicants should have an earned Ph.D. degree and demonstrated achievements in both research and teaching. The teaching language at SUSTech is bilingual, either English or Putonghua. It is perfectly acceptable to use English in all lectures, assignments, exams. In fact, our existing faculty members include several non-Chinese speaking professors.

As a State-level innovative city, Shenzhen has identified innovation as the key strategy for its development. It is home to some of China's most successful high-tech companies, such as Huawei and Tencent. SUSTech considers entrepreneurship as one of the main directions of the university. Strong supports will be provided to possible new initiatives. SUSTech encourages candidates with experience in entrepreneurship to apply.

The Department of Computer Science and Engineering at SUSTech was founded in 2016. It has 17 professors, all of whom hold doctoral degrees or have years of experience in overseas universities. Among them, two were elected into the "1000 Talents" Program in China; three are IEEE fellows; one IET fellow. The department is expected to grow to 50 tenure track faculty members eventually, in addition to teaching-only professors and research-only professors.

SUSTech is a pioneer in higher education reform in China. The mission of the University is to become a globally recognized research university which emphasizes academic excellence and promotes innovation, creativity and entre-

preneurship. Set on five hundred acres of wooded landscape in the picturesque Nanshan (South Mountain) area, the campus offers an ideal environment for learning and research. SUSTech is committed to increase the diversity of its faculty, and has a range of family-friendly policies in place. The university offers competitive salaries and fringe benefits including medical insurance, retirement and housing subsidy, which are among the best in China. Salary and rank will commensurate with qualifications and experience. More information can be found at <http://talent.sustc.edu.cn/en>

We provide some of the best start-up packages in the sector to our faculty members, including one PhD studentship per year, in addition to a significant amount of start-up funding.

To apply, please provide a cover letter identifying the primary area of research, curriculum vitae, and research and teaching statements, and forward them to cshire@sustc.edu.cn.

Swarthmore College
Computer Science Department
Visiting Faculty Positions in Computer Science

The Computer Science Department invites applications for multiple visiting positions at the rank of Assistant Professor to begin Fall semester 2018.

For the visiting position, strong applicants in any area will be considered. Priority will be given to complete applications received by February 15, 2018.

Applications will continue to be accepted after these dates until the positions are filled.

The Computer Science Department currently has eight tenure-track faculty and four visiting faculty. Faculty teach introductory courses as well as advanced courses in their research areas. We have grown significantly in both faculty and students in the last five years. Presently, we are one of the most popular majors at the College and expect to have over 70 Computer Science majors graduating this year.

QUALIFICATIONS: Applicants must have teaching experience and should be comfortable teaching a wide range of courses at the introductory and intermediate level. Candidates should additionally have a strong commitment to involving undergraduates in their research. A Ph.D. in Computer Science at or near the time of appointment is required. The strongest candidates will be expected to demonstrate a commitment to creative teaching and an active research program that speaks to and motivates undergraduates from diverse backgrounds.

APPLICATION INSTRUCTIONS: Applications should include a cover letter, vita, teaching statement, research statement, and three letters of reference, at least one (preferably two) of which should speak to the candidate's teaching ability. In your cover letter, please briefly describe your current research agenda; what would be attractive to you about teaching diverse students in a liberal arts college environment; and what background, experience, or interests are likely to make you a strong teacher of Swarthmore College students.

This institution is using Interfolio's Faculty Search to conduct this search. Applicants to this position receive a free Dossier account

and can send all application materials, including confidential letters of recommendation, free of charge. To apply, visit <https://apply.interfolio.com/45234>.

Swarthmore College actively seeks and welcomes applications from candidates with exceptional qualifications, particularly those with demonstrable commitments to a more inclusive society and world. Swarthmore College is an Equal Opportunity Employer. Women and minorities are encouraged to apply.

Toyota Technological Institute
Principal Professor

Toyota Technological Institute has one opening for "Principal Professor" positions in its School of Engineering. For more information, please refer to the following website: <http://www.toyota-ti.ac.jp/english/employment/index.html>.

Research fields: *Science and technology for advanced instrumentation and/or information processing*

Examples: New devices and systems for advanced information processing, communication, and/or sensing; Leading instrumentation technologies for ultra-sensitive measurements and/or bio-medical studies and diagnosis; Science and technology of cyber-physical systems.

Qualifications: A successful candidate must have a Ph. D. degree or the equivalent in a relevant field; he/she must possess outstanding competence to promote world-class research program(s) as well as to conduct excellent teaching and research supervision for graduate and undergraduate students, so as to fulfill his/her mission as a superb leader in research and education.

Positions: Principal Professor

The "Principal Professor" will serve as the head of a "unit laboratory," that consists of the Principal Professor, one associate professor, and three post-doctoral fellows. A start-up grant of about one hundred million Japanese yen (ca. one million US dollars) is available. In addition, a research budget of about ten million Japanese yen (ca. one hundred thousand US dollars) will be given each year to promote research programs for a period of five years. At the end of this five-year term, the principal professor will be given a formal evaluation.

Number of Positions Available: One

Start date: April 1, 2019 or on the date of the earliest convenience

Documents:

1. A curriculum vitae
2. A list of publications
3. Copies of 5 representative papers
4. An outline of research and educational accomplishments (about 3-pages)
5. A future plan of educational and research activities (about 3-pages)
6. Names of two references, including phone numbers and e-mail addresses
7. An application form (available on our website)

Deadline: May 15, 2018

Inquiries:

Search Committee Chair, Dr. Kazuo Hotate, Vice President & Professor.

(Phone) +81-52-809-1821 (E-mail) hotate-koubo@toyota-ti.ac.jp

The above documentation should be sent to:
Mr. Masashi Hisamoto
Toyota Technological Institute
2-12-1, Hisakata, Tempaku-ku, Nagoya, 468-8511 Japan

Please write "Application for Principal Professor" on the envelope.

The application documents will not be returned.

Western Michigan University

Assistant/Associate Professor in Computer Science

Applications are invited for a tenure-track position at the assistant or associate professor level in the area of applied information security in the Department of Computer Science at Western Michigan University (Kalamazoo, MI) starting August 2018 or January 2019.

Applicants must have a Ph.D. in Computer Science or a closely related field. We are looking for candidates with expertise in applied information security to support our new M.S. in Information Security. The program is offered fully online and in cooperation with the Department of Business Information Systems.

Successful candidates will be capable of establishing an active research program leading to funding, supervising graduate students, and teaching courses at both the undergraduate and graduate levels in information security. Other duties include development of undergraduate and graduate courses, advising and service at the University, College, Department and professional society levels.

Application screening will start immediately and the position will remain open until filled. Successful candidates must earn their Ph.D. degree by the time of employment.

The Department has 260 undergraduates, 50 M.S. students and 45 Ph.D. students. Current active research areas include security, privacy, networks, embedded systems/internet of things, compilers, computational biology, massive data analytics, scientific computing, parallel computing, formal verification, parallel debugging, and data mining. More information regarding Western Michigan University, the College of Engineering and Applied Sciences and the Department of Computer Science are available at <http://www.wmich.edu>, <http://www.wmich.edu/engineer>, and <http://wmich.edu/cs>, respectively.

The Carnegie Foundation for the Advancement of Teaching has placed WMU among the 76 public institutions in the nation designated as research universities with high research activity.

WMU is an Equal Opportunity/Affirmative Action Employer. Minorities, women, veterans, individuals with disabilities and all other qualified individuals are encouraged to apply.

To do so, please visit: <http://wmich.edu/hr/jobs> and provide a cover letter, curriculum vitae, statement of research goals, teaching statement, and names and contact information of at least three references.

[CONTINUED FROM P. 120]

In one sense, it's very practical. Deep learning has been successful not just because it works well, but also because it automates part of the process of building and designing intelligent systems. In the old days, everything was manual; you had to find a way to express all of human knowledge in a set of rules, which turns out to be extremely complicated. Even in the more traditional realm of machine learning, part of the system was trained, but most of it was still done by hand, so for classical computer vision systems, you had to design a way to pre-process the image to get it into a form that your learning algorithm could digest.

With deep learning, on the other hand, you can train an entire system more or less from end to end.

Yes, but you need a lot of labeled data to do it, which limits the number of applications and the power of the system, because it can only learn whatever knowledge is present within your labeled datasets. The more long-term reason for trying to train or pre-train a learning system on unlabeled data is that, as you said, animals and humans build models of the world mostly by observation, and we'd like machines to do that as well, because accumulating massive amounts of knowledge about the world is the only they will eventually acquire a certain level of common sense.

What about adversarial training, in which a set of machines learn together by pursuing competing goals?

This is an idea that popped up a few years ago in Yoshua Bengio's lab with Ian Goodfellow, one of his students at the time. One important application is predictions. If you build a self-driving car or any other kind of system, you'd like that system to be able to predict what's going to happen next—to simulate the world and see what a particular sequence of actions will produce without actually doing it. That would allow it to anticipate things and act accordingly, perhaps to correct something or plan in advance.

How does adversarial training address the problem of prediction in the presence of uncertainty?

When I show you a segment of a video and I ask what happens next, you might be able to predict to some extent, but not exactly; there are probably several different outcomes that are possible. So when you train a system to predict the future, and there are several possible futures, the system takes an average of all the possibilities, and that's not a good prediction.

Adversarial training allows us to train a system where there are multiple correct outputs by asking it to make a prediction, then telling it what should have been predicted. One of the central ideas behind this is that you train two neural networks simultaneously; there is one neural net that does the prediction and there's a second neural net that essentially assesses whether the prediction of the first neural net looks probable or not.

You recently helped found the Partnership on AI, which aims to develop and share best practices and provide a platform for public discussion.

There are questions related to the deployment and perception of AI within the public and government, questions about the ethics of testing, reliability, and many other things that we thought went beyond a single company.

Thanks to rapid advances in the field, many of these questions are coming up very quickly. It seems like there's a lot of excitement, but also a lot of apprehension in the public about where AI is headed.

Humans make decisions under what's called bounded rationality. We are very limited in the time and effort we can spend on any decision. We are biased, and we have to use our bias because that makes us more efficient, though it also makes us less accurate. To reduce bias in decisions, it's better to use machines. That said, you need to apply AI in ways that are not biased, and there are techniques being developed that will allow people to make sure that the decisions made by AI systems have as little bias as possible.

Leah Hoffmann is a technology writer based in Piermont, NY, USA.

© 2018 ACM 0001-0782/18/3 \$15.00

ACM Journal on Computing and Cultural Heritage



ACM JOCCH publishes papers of significant and lasting value in all areas relating to the use of ICT in support of Cultural Heritage, seeking to combine the best of computing science with real attention to any aspect of the cultural heritage sector.



For further information or to submit your manuscript, visit jocch.acm.org

Q&A

The Network Effect

The developer of convolutional neural networks looks at their impact, today and in the long run.

DEEP LEARNING MIGHT be a booming field these days, but few people remember its time in the intellectual wilderness better than Yann LeCun, director of Facebook Artificial Intelligence Research (FAIR) and a part-time professor at New York University. LeCun developed convolutional neural networks while a researcher at Bell Laboratories in the late 1980s. Now, the group he leads at Facebook is using them to improve computer vision, to make predictions in the face of uncertainty, and even to understand natural language.

Your work at FAIR ranges from long-term theoretical research to applications that have real product impact.

We were founded with the idea of making scientific and technological progress, but I don't think the Facebook leadership expected quick results. In fact, many things have had a fairly continuous product impact. In the application domain, our group works on things like text understanding, translation, computer vision, image understanding, video understanding, and speech recognition. There are also more esoteric things that have had an impact, like large-scale embedding.

This is the idea of associating every object with a vector.

Yes. You describe every object on Facebook with a list of numbers, whether it's a post, news item, photo, comment, or user. Then, you use operations between vectors to see if, say, two images are similar, or if a person is likely to be interested in a certain piece



of content, or if two people are likely to be friends with one another.

What are some of the things going on at FAIR that most interest or excite you?

It's all interesting! But I'm personally interested in a few things.

One is marrying reasoning with learning. A lot of learning has to do with perceptions, which are relatively simple things that people can do without thinking too much. But we haven't yet found good recipes for training systems to do tasks that require a little bit of reasoning. There is some work in that direction, but it's not where we want it.

Another area that interests me is unsupervised learning—teaching machines to learn by observing the world, say by watching videos or looking at images without being told what objects are in these images.

And the last thing would be autonomous AI systems whose behavior is not directly controlled by a person. In other words, they are designed not just to do one particular task, but to make decisions and adapt to different circumstances on their own.

How does the interplay work between research and product?

There's a group called Applied Machine Learning, or AML, that works closely with FAIR and is a bit more on the application side of things. That group did not exist when I joined Facebook, but I pushed for its creation, because I saw this kind of relationship work very well at AT&T. Then AML became a victim of its own success. There was so much demand within the company for the platforms they were developing, which basically enabled all kinds of groups within Facebook to use machine learning in their products, that they ended up moving away from FAIR.

Recently we reorganized this a little bit. A lot of the AI capability is now being moved to the product groups, and AML is refocusing on the advanced development of things that are close to research. In certain areas like computer vision, there is a very, very tight collaboration, and things go back and forth really quickly. In other areas that are more disruptive or for which there is no obvious product, it's more like, 'let us work on it for a few years first'.

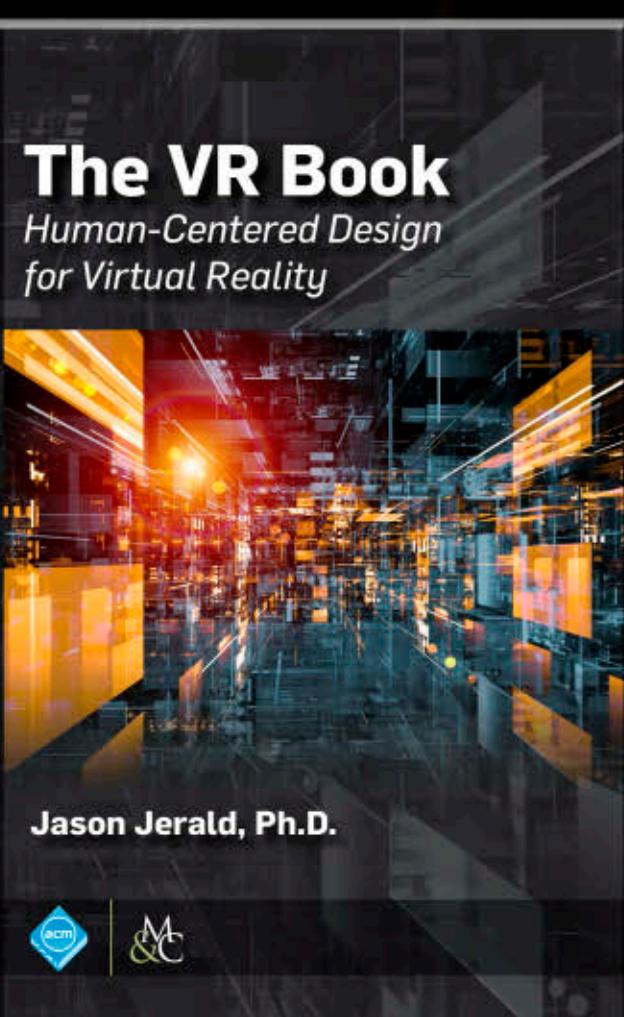
Let's talk about unsupervised learning, which, as you point out elsewhere, is much closer to the way that humans actually learn.

[CONTINUED ON P. 119]

Jason Jerald, PhD

The VR Book

Human-Centered Design for Virtual Reality



Dr. Jerald has recognized a great need in our community and filled it. The VR Book is a scholarly and comprehensive treatment of the user interface dynamics surrounding the development and application of virtual reality. I have made it required reading for my students and research colleagues. Well done!"

- Prof. Tom Furness, University of Washington, VR Pioneer



ISBN: 978-1-970001-12-9 DOI: 10.1145/2792790
<http://books.acm.org>
<http://www.morganclaypoolpublishers.com/vr>

2018 Artificial Intelligence · Blockchain · Cloud

— CREATING THE FUTURE —

June 25 - June 30, Seattle, USA

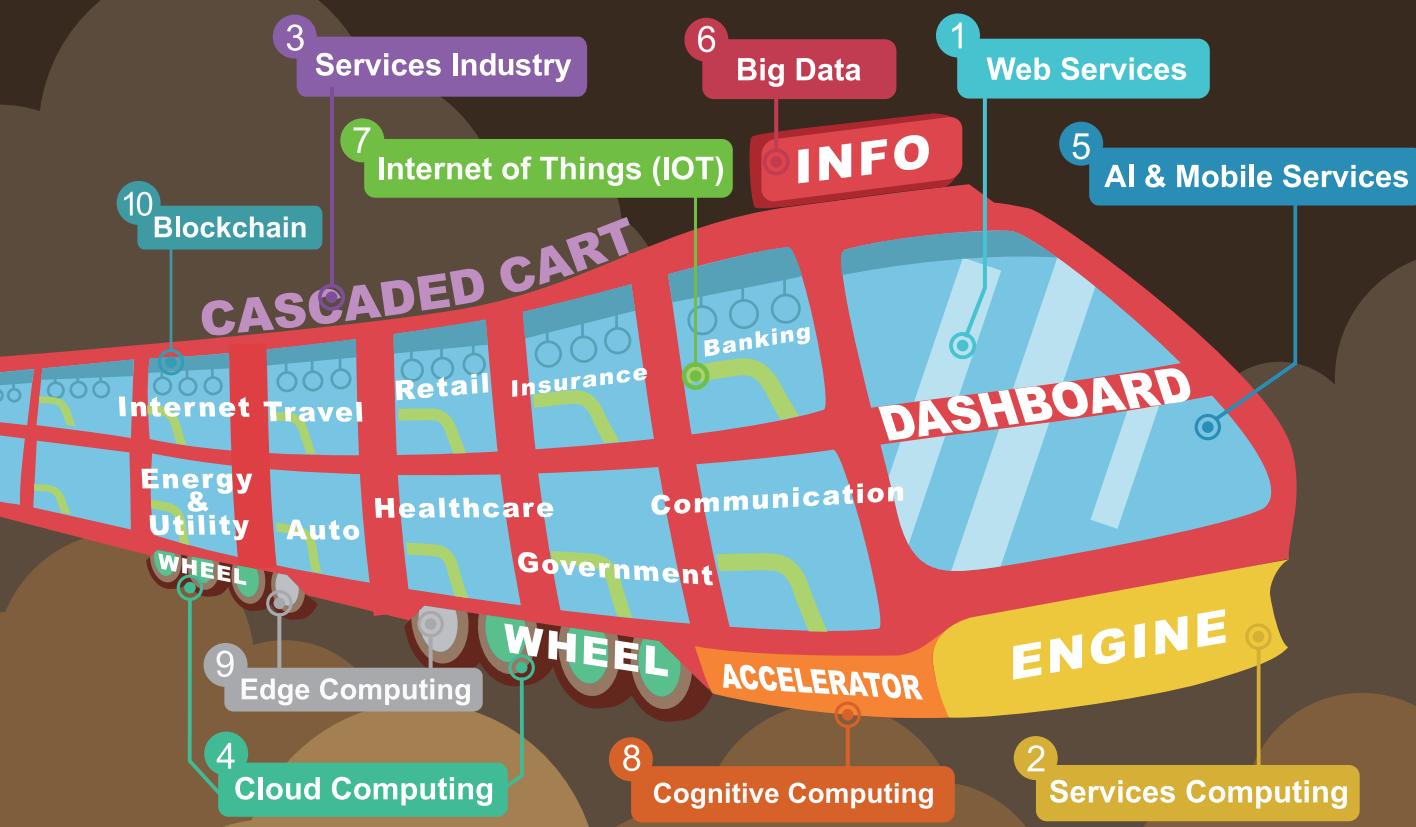
CELEBRATING THE 25th GATHERING OF ICWS



SCF

SERVICES
CONFERENCE
FEDERATION

- 1 The 25th International Conference on Web Services (**ICWS 2018**)
- 2 The 15th International Conference on Services Computing (**SCC 2018**)
- 3 The 14th World Congress on Services (**SERVICES 2018**)
- 4 The 11th International Conference on Cloud Computing (**CLOUD 2018**)
- 5 The 7th International Conference on AI & Mobile Services (**AIMS 2018**)
- 6 The 7th International Congress on Big Data (**BigData Congress 2018**)
- 7 The 3rd International Conference on Internet of Things (**ICIOT 2018**)
- 8 The 2nd International Conference on Cognitive Computing (**ICCC 2018**)
- 9 The 2nd International Conference on Edge Computing (**EDGE 2018**)
- 10 The 1st International Conference on Blockchain (**ICBC 2018**)



Submission Deadlines

2/6/2018: ICWS 2018 (<http://icws.org>)
 2/21/2018: SCC 2018 (<http://theSCC.org>)
 2/25/2018: SERVICES 2018 (<http://ServicesCongress.org>)
 2/6/2018: CLOUD 2018 (<http://theCloudComputing.org>)
 2/21/2018: AIMS 2018 (<http://ai1000.org>)

2/28/2018: BigData Congress 2018 (<http://BigDataCongress.org>)
 3/17/2018: ICIOT 2018 (<http://iciot.org>)
 3/17/2018: ICCC 2018 (<http://theCognitiveComputing.org>)
 3/17/2018: EDGE 2018 (<http://theEdgeComputing.org>)
 3/17/2018: ICBC 2018 (<http://Blockchain1000.org>)

Contact: ZHANGLJ@ACM.ORG

Dr. Liang-Jie (LJ) Zhang

Steering Committee Chair

