

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

05/2016 VOL.59 NO.05

A Survey of Robotic Musicianship

Coding as Sport
The Moral Imperative of AI
Increasing the Security
of Smart Buildings
ACM's 2016 General Election

Association for
Computing Machinery



Applicative 2016

June 1 – 2, 2016
New York City

APPLICATIVE 2016 will bring together practitioners and researchers to share the latest emerging technologies and trends in software development.

The conference consists of two tracks:

APPLICATION DEVELOPMENT will feature speakers from leading technology companies such as Google and Facebook, talking about how they are applying new technologies to the products they deliver. The track covers topics such as reactive programming, micro-services, single-page application frameworks, and other approaches that will help you build more robust applications and do it more quickly.

SYSTEMS SOFTWARE will explore topics that enable systems-level practitioners to build better software for the modern world. The speakers are involved in the design, implementation and support of novel technologies and low-level software supporting some of today's most demanding workloads.

For more information about the conference and how to register, please visit:

<http://applicative.acm.org>



Association for
Computing Machinery



THE ACM A. M. TURING AWARD

by the community ♦ from the community ♦ for the community



ACM and Google congratulate

WHITFIELD DIFFIE and MARTIN E. HELLMAN

For the conception and first effective realization of public-key cryptography. The idea of a public-key cryptosystem was a major conceptual breakthrough that continues to stimulate research to this day and without it today's rapid growth of electronic commerce would not have been possible.



"Public key cryptography is fundamental for our industry. The ability to protect private data rests on protocols for confirming an owner's identity and for ensuring the integrity and confidentiality of communications. These widely used protocols were made possible through the ideas and methods pioneered by Diffie and Hellman."

Andrei Broder
Google Distinguished Scientist
Google Inc.

Google™

For more information see <http://research.google.com/>

Departments

- 5 **Editor's Letter**
The Moral Imperative of Artificial Intelligence
By Moshe Y. Vardi
-
- 7 **Cerf's Up**
The IANA Transition
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Why All Writs Is a Trojan Horse
-
- 11 **ACM's 2016 General Election**
-
- 24 **BLOG@CACM**
Bringing Computer Science to U.S. Schools, State by State
Mark Guzdial considers the crucial role of states in Computer Science for All.
-
- 49 **Calendar**
-
- 132 **Careers**

Last Byte

- 136 **Future Tense**
Becoming a Multi-Planet Species
Imagine hyper-realistic virtual space exploration limited only by the data we are able to collect.
By Louis Friedman

News



- 26 **Silicon Photonics: Ready to Go the Distance?**
Processes for making CMOS chips are adapted for optical components.
By Don Monroe
-
- 29 **Cybersecurity Gets Smart**
Researchers aim to apply artificial intelligence and machine-learning methods to take cybersecurity to a new, higher, and better level.
By Samuel Greengard
-
- 32 **Coding as Sport**
Programming competitions help identify and reward those who excel in coding.
By Keith Kirkpatrick

Viewpoints

- 36 **Law and Technology**
The Internet of Things We Don't Own?
Who will control the 'ordinary pursuits of life' in the digital economy?
By Jason Schultz
-
- 39 **Education**
Beyond Blocks: Syntax and Semantics
How the future of general-purpose programming tools could include blocks-based structured editing, and how we should study students transitioning to text-based programming tools.
By R. Benjamin Shapiro and Matthew Ahrens
-
- 42 **Economic and Business Dimensions**
Do Patent Commons and Standards-Setting Organizations Help Navigate Patent Thickets?
Examining the institutions underlying the patent system for information and communication technologies.
By Wen Wen and Chris Forman
-
- 44 **Historical Reflections**
Preserving Hybrid Objects
Brutal lessons from contemporary art.
By David P. Anderson
-
- 47 **Viewpoint**
How to Increase the Security of Smart Buildings?
Surveying unresolved security problems for automated buildings.
By Steffen Wendzel



Association for Computing Machinery
Advancing Computing as a Science & Profession

Practice



58

- 50 **Borg, Omega, and Kubernetes**
Lessons learned from three container-management systems over a decade.

By Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes

- 58 **Delegation as Art**
Be someone who makes everyone else better.

By Kate Matsudaira

- 61 **Use-Case 2.0**
The hub of software development.

By Ivar Jacobson, Ian Spence, and Brian Kerr



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

Contributed Articles



78

- 70 **The Challenges of Partially Automated Driving**
Car automation promises to free our hands from the steering wheel but might demand more from our minds.

By Stephen M. Casner, Edwin L. Hutchins, and Don Norman

- 78 **Parallel Graph Analytics**
Data-centric abstractions and execution strategies are needed to exploit parallelism in large-scale graph analytics.

By Andrew Lenharth, Donald Nguyen, and Keshav Pingali



Watch the authors discuss their work in this exclusive *Communications* video.
<http://cacm.acm.org/videos/parallel-graph-analytics>

- 88 **Static Presentation Consistency Issues in Smartphone Mapping Apps**
Comparing smartphone mapping apps leads to unexpected surprises.

By Hanan Samet, Sarana Nutanong, and Brendan C. Fruin

Review Articles

- 100 **A Survey of Robotic Musicianship**
Reviewing the technologies that enable robot musicians to jam.

By Mason Bretan and Gil Weinberg



Watch the authors discuss their work in this exclusive *Communications* video.
<http://cacm.acm.org/videos/a-survey-of-robotic-musicianship>

Research Highlights

- 112 **Technical Perspective**
A Breakthrough in Software Obfuscation

By Boaz Barak

- 113 **Hiding Secrets in Software: A Cryptographic Approach to Program Obfuscation**

By Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters

- 121 **Technical Perspective**
Software Is Natural

By Gail C. Murphy

- 122 **On the Naturalness of Software**

By Abram Hindle, Earl T. Barr, Mark Gabel, Zhendong Su, and Premkumar Devanbu

**About the Cover:**

This month's cover story on robotic musicianship (p. 100) goes beyond machines that produce sound. Mason Bretan and Gil Weinberg examine cutting-edge robotics that generate and showcase musicality and interact with human musicians. Cover photo illustration by Barry Downard.



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

Executive Director and CEO
Bobby Schnabel
Deputy Executive Director and COO
Patricia Ryan
Director, Office of Information Systems
Wayne Graves
Director, Office of Financial Services
Darren Ramdin
Director, Office of SIG Services
Donna Cappel
Director, Office of Publications
Bernard Rous
Director, Office of Group Publishing
Scott E. Delman

ACM COUNCIL

President
Alexander L. Wolf
Vice-President
Vicki L. Hanson
Secretary/Treasurer
Erik Altman
Past President
Vinton G. Cerf
Chair, SGB Board
Patrick Madden
Co-Chairs, Publications Board
Jack Davidson and Joseph Konstan
Members-at-Large
Eric Allman; Ricardo Baeza-Yates;
Cherri Pancake; Radia Perlman;
Mary Lou Soffa; Eugene Spafford;
Per Stenström
SGB Council Representatives
Paul Beame; Jenna Neefe Matthews;
Barbara Boucher Owens

BOARD CHAIRS

Education Board
Mehran Sahami and Jane Chu Prey
Practitioners Board
George Neville-Niel

REGIONAL COUNCIL CHAIRS

ACM Europe Council
Dame Professor Wendy Hall
ACM India Council
Srinivas Padmanabhuni
ACM China Council
Jiaquan Sun

PUBLICATIONS BOARD

Co-Chairs
Jack Davidson; Joseph Konstan
Board Members
Ronald F. Boisvert; Anne Condon;
Nikil Dutt; Roch Guerrin; Carol Hutchins;
Yannis Ioannidis; Catherine McGeoch;
M. Tamer Ozsu; Mary Lou Soffa; Alex Wade;
Keith Webster

ACM U.S. Public Policy Office

Renee Dopplick, Director
1828 L Street, N.W., Suite 800
Washington, DC 20036 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Mark R. Nelson, 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 GROUP PUBLISHING
Scott E. Delman
cacm-publisher@cacm.acm.org

Executive Editor

Diane Crawford
Managing Editor
Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Larry Fisher

Web Editor

David Roman

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Director

Mia Angelica Balaquiot

Designer

Iwona Usakiewicz

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Publications Assistant

Juliet Chance

Columnists

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

CONTACT POINTS

Copyright permission
permissions@cacm.acm.org

Calendar items
calendar@cacm.acm.org

Change of address
acmhelp@cacm.acm.org

Letters to the Editor
letters@cacm.acm.org

WEBSITE

http://cacm.acm.org

AUTHOR GUIDELINES

http://cacm.acm.org/

ACM ADVERTISING DEPARTMENT

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

Director of Media Sales

Jennifer Ruzicka
jen.ruzicka@hq.acm.org

Media Kit acmm mediasales@cacm.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

Moshe Y. Vardi
eic@cacm.acm.org

NEWS

Co-Chairs

William Pulleyblank and Marc Snir

Board Members

Mei Kobayashi; Kurt Mehloth;
Michael Mitzenmacher; Rajeev Rastogi

VIEWPOINTS

Co-Chairs

Tim Finin; Susanne E. Hambrusch;
John Leslie King

Board Members

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



PRACTICE

Co-Chair

Stephen Bourne

Board Members

Eric Allman; Peter Bailis; Terry Coatta;
Stuart Feldman; Benjamin Fried;
Pat Hanrahan; Tom Killalea; Tom Limoncelli;
Kate Matsudaira; Marshall Kirk McKusick;
George Neville-Niel; Theo Schlossnagle;
Jim Waldo

The Practice section of the CACM
Editorial Board also serves as
the Editorial Board of *ACM Queue*.

CONTRIBUTED ARTICLES

Co-Chairs

Andrew Chien and James Larus

Board Members

William Aiello; Robert Austin; Elisa Bertino;
Gilles Brassard; Kim Bruce; Alan Bundy;
Peter Buneman; Peter Druschel; Carlo Ghezzi;
Carl Gutwin; Yannis Ioannidis;
Gal A. Kaminka; James Larus; Igor Markov;
Gail C. Murphy; Bernhard Nebel;
Lionel M. Ni; Kenton O'Hara; Sriram Rajamani;
Marie-Christine Rousset; Avi Rubin;
Krishan Sabnani; Ron Shamir; Yoav
Shoham; Larry Snyder; Michael Vitale;
Wolfgang Wahlster; Hannes Werthner;
Reinhard Wilhelm

RESEARCH HIGHLIGHTS

Co-Chairs

Azer Bestavros and Gregory Morrisett

Board Members

Martin Abadi; Amr El Abbadi; Sanjeev Arora;
Nina Balcan; Dan Boneh; Andrei Broder;
Doug Burger; Stuart K. Card; Jeff Chase;
Jon Crowcroft; Sandhya Dwaekadas;
Matt Dwyer; Alon Halevy; Norm Jouppi;
Andrew B. Kahng; Sven Koenig; Xavier Leroy;
Steve Marschner; Kobbi Nissim;
Steve Seitz; Guy Steele, Jr.; David Wagner;
Margaret H. Wright; Andreas Zeller

WEB

Chair

James Landay

Board Members

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

ACM Copyright Notice

Copyright © 2016 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@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 U.S.A.



Association for
Computing Machinery





DOI:10.1145/2903530

Moshe Y. Vardi

The Moral Imperative of Artificial Intelligence

The big news on March 12 of this year was of the Go-playing AI-system AlphaGo securing victory against 18-time world champion Lee Se-dol by winning the third straight game

of a five-game match in Seoul, Korea. (AlphaGo is a program developed by DeepMind, a British AI company acquired by Google two years ago.) After Deep Blue's victory against chess world champion Gary Kasparov in 1997, the game of Go was the next grand challenge for game-playing artificial intelligence. Go has defied the brute-force methods in game-tree search that worked so successfully in chess. In 2012, *Communications* published a Research Highlight article by Sylvain Gelly et al. on computer Go, which reported that "Programs based on Monte-Carlo tree search now play at human-master levels and are beginning to challenge top professional players." AlphaGo combines tree-search techniques with search-space reduction techniques that use deep learning. Its victory is a stunning achievement and another milestone in the inexorable march of AI research.

By using deep-learning techniques to prune the search tree, AlphaGo can be said to augment brute-force search with "intuition," which it has developed by playing numerous games against itself. (Google said AlphaGo does not use Lee's games as its training data.) By relying on learned "intuition," AlphaGo is able to overcome the so-called *Polanyi's Paradox*. The philosopher Michael Polanyi observed in 1966, "We can know more than we can tell ... The skill of a driver cannot be replaced by a thorough schooling in the theory of the mo-

torcar." Some labor economists have viewed Polanyi's Paradox as a major barrier for AI, arguing it implies a limit on its potential to automate human jobs. AlphaGo's victory demonstrates machine learning provides a path around that barrier.

Indeed, the automation of driving has been a major challenge for AI research over the past decade. In 2004, economists argued driving was unlikely to be automated in the near future due to Polanyi's Paradox. A year later, a Stanford autonomous vehicle won a DARPA Grand Challenge by driving over 100 miles along an unrehearsed desert trail. Now, more than a decade later, both technology companies and car companies vigorously pursue the automation of driving. I expect the technical challenges to be resolved in the coming decade.

In 1843, Ada Lovelace, well known for her work on Charles Babbage's Analytical Engine, wrote to Babbage that she wished to see computing technology developed for "the most effective use of mankind." It is difficult for me to think of any computing technology other than automated driving that can be deployed in a decade or two and with such benefit for humanity. About 1.25 million people worldwide die from car accidents every year. Over 90% of these accidents are caused by human error. By automating driving, we could save over a million lives a year, as well as avoid countless injuries.

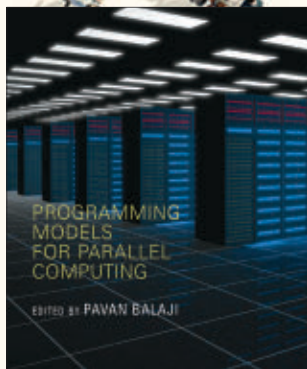
At the same time, the automation of driving would have a huge disruptive effect on the global economy. Existing industries will shrivel, and whole new industries will rise. In the U.S., close to 10% of all jobs involve operating a vehicle and we can expect to see the majority of these jobs disappear. The human cost of such a profound change cannot be underestimated. As a precedent we can see what happened to U.S. manufacturing over the past 35 years. While manufacturing output in constant dollars is at an all-time high, manufacturing employment peaked around 1980 and is today lower than it was in 1947. The disappearance of millions of jobs due to automation may explain a recent, rather shocking, finding by economists Angus Deaton, winner of the 2015 Nobel Memorial Prize in Economic Science, and Anne Case, that mortality for white middle-aged Americans has been increasing over the past 25 years, due to an epidemic of suicides and afflictions stemming from substance abuse.

Thus, the automation of driving would be hugely beneficial, saving lives and preventing injuries on a massive scale. At the same time, it would have a profoundly adverse impact on the labor market. In the balance, life saving and injury prevention must take precedence, and we have a moral imperative to develop and deploy automated driving. The solution to the labor problem will not be technical, but sociopolitical. As computing professionals, we also have a moral imperative to acknowledge the adverse societal consequences of the technology we develop and to engage with social scientists to find ways to address these consequences.

Follow me on Facebook, Google+, and Twitter.

Moshe Y. Vardi, EDITOR-IN-CHIEF

Copyright held by author.

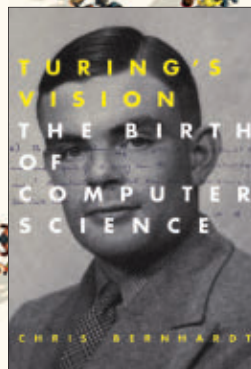


PROGRAMMING MODELS FOR PARALLEL COMPUTING

edited by Pavan Balaji

An overview of the most prominent contemporary parallel processing programming models, written in a unique tutorial style.

Scientific and Engineering Computation series



TURING'S VISION

The Birth of Computer Science

Chris Bernhardt

"A fascinating account of Alan Turing's epic research paper, which kicked off the entire computer revolution."

—Ian Stewart, author of *In Pursuit of the Unknown: 17 Equations That Changed the World*



CROWDSOURCED HEALTH

How What You Do on the Internet Will Improve Medicine

Elad Yom-Tov

"This book tells us the 'good news' about large-scale data collected over the Internet. . . In clear and simple prose, Elad Yom-Tov makes a compelling case for the magic of crowdsourcing health."

—Oren Etzioni, CEO of the Allen Institute for Artificial Intelligence



CLOUD COMPUTING

Nayan B. Ruparelia

Why cloud computing represents a paradigm shift for business, and how business users can best take advantage of cloud services.

The MIT Press Essential Knowledge series



mitpress.mit.edu

World-Renowned Journals from ACM

ACM publishes over 50 magazines and journals that cover an array of established as well as emerging areas of the computing field. IT professionals worldwide depend on ACM's publications to keep them abreast of the latest technological developments and industry news in a timely, comprehensive manner of the highest quality and integrity. For a complete listing of ACM's leading magazines & journals, including our renowned Transaction Series, please visit the ACM publications homepage: www.acm.org/pubs.

ACM Transactions on Interactive Intelligent Systems



ACM Transactions on Interactive Intelligent Systems (TIIS). This quarterly journal publishes papers on research encompassing the design, realization, or evaluation of interactive systems incorporating some form of machine intelligence.

ACM Transactions on Computation Theory



ACM Transactions on Computation Theory (ToCT). This quarterly peer-reviewed journal has an emphasis on computational complexity, foundations of cryptography and other computation-based topics in theoretical computer science.

PLEASE CONTACT ACM MEMBER SERVICES TO PLACE AN ORDER
 Phone: 1.800.342.6626 (U.S. and Canada)
 +1.212.626.0500 (Global)
 Fax: +1.212.944.1318
 (Hours: 8:30am–4:30pm, Eastern Time)
 Email: acmhelp@acm.org
 Mail: ACM Member Services
 General Post Office
 PO Box 30777
 New York, NY 10087-0777 USA



Association for Computing Machinery

Advancing Computing as a Science & Profession

www.acm.org/pubs



Vinton G. Cerf

DOI:10.1145/2908732

The IANA Transition

This month I would like to explore the notable proposal sent in March 2016 by the Internet Corporation for Assigned Names and Numbers (ICANN) to the U.S.

Department of Commerce, National Telecommunication and Information Agency (NTIA) to end the long-standing contractual relationship between ICANN and NTIA for the conduct of the Internet Assigned Numbers Authority functions (“IANA functions”). ICANN was formed in the fall of 1998 in response to a White House “White Paper” issued by Ira Magaziner, then a senior advisor for policy to President Bill Clinton. ICANN would undertake to form a private sector entity to carry out the coordinated assignment of Internet domain names, Internet addresses, and the maintenance of parameter registries needed for the operation of the suite of protocols used in the Internet.

These functions had been managed by Jonathan Postel acting as the IANA at USC Information Sciences Institute (and other earlier institutions where Postel had worked) under various government contracts. By 1996, the Internet was experiencing its so-called “dot boom” and the potential scale and liabilities of carrying out the IANA functions led to a serious effort to institutionalize the operation. For lack of space, I will leave out two years of community debate and fast-forward to the creation of ICANN to fulfill these functions. ICANN was conceived as a multi-stakeholder organization drawing on input from the private sector, civil society, governments of the world, and the technical community for the development of policy for the IANA functions and for the coordination of the multiple parties having a role in managing these unique identifiers and parameters.

In 1998, many organizations were involved in the evolution and operation of the Internet, its Domain Name System (DNS), Internet address allocation, and standards development. The Internet Society, founded in 1991, housed the standards-oriented Internet Architecture Board and the Internet Engineering Task Force. There were then three Regional Internet Registries (RIRs) for Internet address allocation—RIPE-NCC, APNIC, and ARIN—and two more to follow later (LACNIC and AFRINIC). There were nominally 13 DNS Root Server operators providing top-level domain name resolution. Verisign generated and distributed the official domain name root zone based on input from IANA and, under the terms of the NTIA/ICANN contract, authorization from NTIA. Many domain name registries and registrars were created to support DNS operation.

The original plan was for ICANN to operate under NTIA oversight for a few years and then operate as an independent organization. In fact, the contractual obligations extended from 1998 to the present. In March 2014, however, NTIA proposed this contractual relationship for the IANA functions should be ended and ICANN be allowed to perform the IANA functions independently. In March 2016, ICANN delivered to NTIA its consolidated proposal from all the constituent parties for the transition from the present contractual relationship to independent operation. The two-year effort leading to this comprehensive proposal was not

without considerable debate among all the parties. Many ideas were surfaced, analyzed, argued over, adopted, adapted, or discarded leading to a consolidated result. The Department of Commerce and the U.S. Congress will be evaluating the proposed new *modus operandi* in the weeks ahead.

Some fears have been voiced that the complex proposal poses risks that authoritarian governments within the ICANN Governmental Advisory Committee (GAC) or through some external means might wrest control of ICANN from its multi-stakeholder constituencies. While the proposal should be evaluated on all its merits, I am persuaded the terms and conditions of the proposed operating practices are well protected against such an outcome. A great many conditions must be satisfied before the more extraordinary powers of the *sole designator* can be exercised. The headquarters of ICANN will remain in the U.S. The many entities that cooperate with ICANN to manage core Internet identifier administration have expressed full support for the proposal.

If I have any trepidation about the proposal it is associated with its general complexity. As the former chairman of ICANN, I am no stranger to the evolution of ICANN’s structure and processes and their relative intricacy. The new proposal adds its own unique aspects to this tendency and it remains to be seen how well the system will work. However, ICANN has shown a remarkable ability to reform and adapt at need and I believe that capacity is preserved under the new proposal. There is still a good deal of work ahead to actually implement what is ultimately approved, but I feel confident this community is capable of achieving a successful outcome. ■

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

Copyright held by author.

Why All Writs Is a Trojan Horse

CITING THE ALL WRITS ACT as a way to give the government the power to compel companies to redesign or reimplement their electronic products to government specifications represents a threat to everyone's civil liberties. This U.S. federal statute, first adopted in 1789 and updated in 1911, today plays a pivotal role in the FBI vs. Apple legal battle in which a federal magistrate in California ordered Apple, at the behest of the FBI, to create and sign a new version of Apple's iOS operating system. In a related court filing, the Justice Department said all information on an electronic device must be accessible to police under court warrant. Does that mean the government has the authority to obtain a court order to compel Apple to redesign its next generation of iPhones so it can break into and read any encrypted information? Moreover, can it prohibit importation of, say, Samsung smartphones for which Samsung has no ability to break in or read encrypted information?

Meanwhile, the Internet of Things (IoT) is becoming pervasive in every aspect of our lives, including insulin pumps, heart pacemakers, TVs with microphones, home video cameras, and even brain implants if a new DARPA project is successful. The Justice Department has proposed security agencies be given the authority to secretly access and take control of any IoT device when ordered by a court or by the executive branch. However, this would make it difficult for a court or even for Congress to prevent them from accessing and controlling large numbers of devices, potentially abusing their court- or executive branch-authorized surveillance authority.

The U.S. Congress should thus prohibit the potential usurpation of power that could result from All Writs, which predates the U.S. Bill of Rights, to require creation of mandatory backdoors in electronic devices. Toward this end, I would like to propose the

following legislative principles to help guide the law in safeguarding our civil liberties. The U.S. government or any of its political subdivisions, including a state or its political subdivisions, may not order or coerce a manufacturer, seller, developer, or provider of computer hardware, software, or device made available to the general public to design, alter, or modify the related security features in order to allow a federal or state agency to obtain information stored in such a device or provide the ability to decrypt information encrypted therein.

This proposal aims to balance the Constitutional requirement to protect citizens' civil liberties and for law enforcement to catch and prosecute criminals, including terrorists. It would uphold the U.S. Constitution's Fifth Amendment right against self-incrimination by prohibiting mandatory IoT backdoors that could provide access to sensitive personal information. At the same time, it would not prohibit access to "distributed encrypted public recording"¹ (such as videos in public places, all financial transactions, and locations of cell phones from cell towers) so all recorded activities except those in personal IoT devices could be subpoenaed.

Carl Hewitt, Palo Alto, CA

Reference

1. Hewitt, C. *Security Without IoT Mandatory Backdoors*. HAL Archives, 2016; <https://hal.archives-ouvertes.fr/hal-01152495>

Without Repeatability, There Is No Computer Science

Christian Collberg and Todd A. Proebsting deserve our gratitude for their article "Repeatability in Computer Systems Research" (Mar. 2016) shining sunlight—the best kind of disinfectant, according to Supreme Court Justice Louis D. Brandeis—on the very real problem of lack of repeatability in computer science research. Without repeatability, there is no real science, something computer science cannot tolerate.

Alex Simonelis, Montréal, Canada

Plenty of Precedent for Banning Autonomous Weapons

Moshe Y. Vardi concluded his Editor's Letter "On Lethal Autonomous Weapons" (Dec. 2015) by saying "Knowledgeable, well-meaning experts are arguing the two sides of the LAWS [lethal autonomous weapons systems] issue. To the best of my knowledge, this is the first time the computing-research community is publicly grappling with an issue of such weight." The debate about lethal autonomous weapons goes back to at least President Ronald Reagan's Strategic Defense Initiative (SDI, or "Star Wars"), which proposed deployment of weapons of various types, including missiles and lasers, that would be controlled by computers and based partly in space. The stated primary purpose of SDI was to shoot down missiles carrying nuclear weapons while in transit, but SDI proposals also considered destroying missile-launch facilities.

The SDI proposal prompted intense worldwide debate about SDI's reliability and risks, including whether life-and-death decisions could and should be entrusted to computer systems. Computer scientists David Parnas, David Bellin, Severo Ornstein, Alan Borning, and others, led by an organization called Computer Professionals for Social Responsibility (CPSR, <http://cpsr.org/>), said computer technology is inherently too unreliable to allow it to make such decisions on its own, without human oversight. CPSR sponsored several debates on the topic and published a book called *Computers in Battle: Will They Work*¹ often cited in today's publications on autonomous weapons.

Gary Chapman,² the editor of the book and CPSR's first executive director, was a key figure in broadening the scope of the debate to include all computerized autonomous weapons. He and his colleagues at the LBJ School of Public Affairs at the University of Texas at Austin and beyond published a number of papers and articles about the reliability, risks, and ethics of LAWS from the late 1980s until Chapman's untimely death at age 58 in 2010.

Regarding the Point/Counterpoint debate “The Case for Banning Killer Robots” by Stephen Goose and Ronald Arkin (also in Dec. 2015), anti-personnel landmines are nothing more, nothing less than autonomous weapons with an extremely simple algorithm: Tread on me and I blow you up. Soldiers and other combatants place them in battle zones, and henceforth they operate without human control. The simplicity of the algorithm does not make them any less “autonomous” than high-tech AI-based weapons. Since well-accepted treaties have banned them internationally, it seems there is a strong precedent for banning computer-controlled LAWS as well.

Jeff Johnson (former chair of CPSR),
San Francisco, CA

References

1. Bellin, D. and Chapman, G. *Computers in Battle: Will They Work*. Harcourt Brace Jovanovich, Boston, MA, 1987.
2. Chapman, G. Thinking about ‘autonomous’ weapons. *CPSR Newsletter 3* (Fall 1987), 11–14.

Author Responds:

I thank Johnson for his instructive letter about the history of the debate on lethal autonomous weapons. In spite of the long history of the subject, the issues raised by strategic-defense weapons, landmines, and newer AI-based LAWS are quite different and should not be conflated.

Moshe Y. Vardi, Editor-in-Chief

Bahrain Deports Masaud Jahromi

I have written several letters to the editor concerning the legal problems and denial of human rights of Masaud Jahromi, formerly chair of the Telecommunication Engineering Department of Ahlia University, Manama, Bahrain, over the past few years, starting with “Justice for Jahromi” (Nov. 2011). Since the last letter, “Bahrain Revokes Masaud Jahromi’s Citizenship” (Apr. 2015), the government of Bahrain has now formally deported him, making Jahromi today a “man without a country.” Here, I briefly review his case and current situation, soliciting public support on his behalf.

In 2011, he was sentenced to four months in prison and fined 500 dinars for participating in “unauthorized rallies” during public protests.

He served his sentence and was eventually restored to his position at Ahlia University. No additional allegations were made or evidence presented that he violated any law since then. Nevertheless, on January 31, 2015, Bahrain revoked his citizenship following an unsubstantiated accusation of “advocating regime change through illegal means.”

Article 15 of the Universal Declaration of Human Rights specifically prohibits arbitrary deprivation of anyone’s nationality. Implementing it, the International Convention on Civil and Political Rights, to which Bahrain is a signatory, protects the right of individuals to due process and their right to free speech. Denial of citizenship without apparent basis in fact or law imposes severe damage on an individual, who consequently becomes stateless and homeless. Moreover, including Jahromi’s name on a list with the names of obvious terrorists serving ISIS abroad has damaged his reputation as an academic.

Despite Jahromi’s attempts to reverse the decision and supportive letters from well-known and respected organizations, including the Committee of Concerned Scientists, of which I am a vice-chair, and other human rights organizations, the Bahraini court has refused to reverse its decision.

In a recent email message, Jahromi wrote, “Currently I hold no nationality. The travel document that I have is valid until March 2017 and in place of nationality field it is indicated ‘Bahrain Resident.’ I have given entrance visa and able to stay in Beirut for 6 months only. Currently, I am staying with my friend and have no job opportunities.” His wife and children were not permitted to leave Bahrain with him.

If any ACM member is able to suggest an opportunity for Jahromi at a college or university, please email me. I will send a copy of his vitae and inform Jahromi.

Jack Minker (minker@cs.umd.edu),
College Park, MD

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.

© 2016 ACM 0001-0782/16/05 \$15.00

Coming Next Month in COMMUNICATIONS

**A Profile of ACM’s
A.M. Turing
Award Recipients
Whitfield Diffie and
Martin E. Hellman**

**An Interview
with Yale Patt**

Improving API Usability

**Randomized Numerical
Linear Algebra**

A Byte Is All We Need

**Computer Science
Should Stay Young**

**Physical Key Extraction
Attacks on PCs**

**Protecting Social Norms
as Confidentiality Wanes**

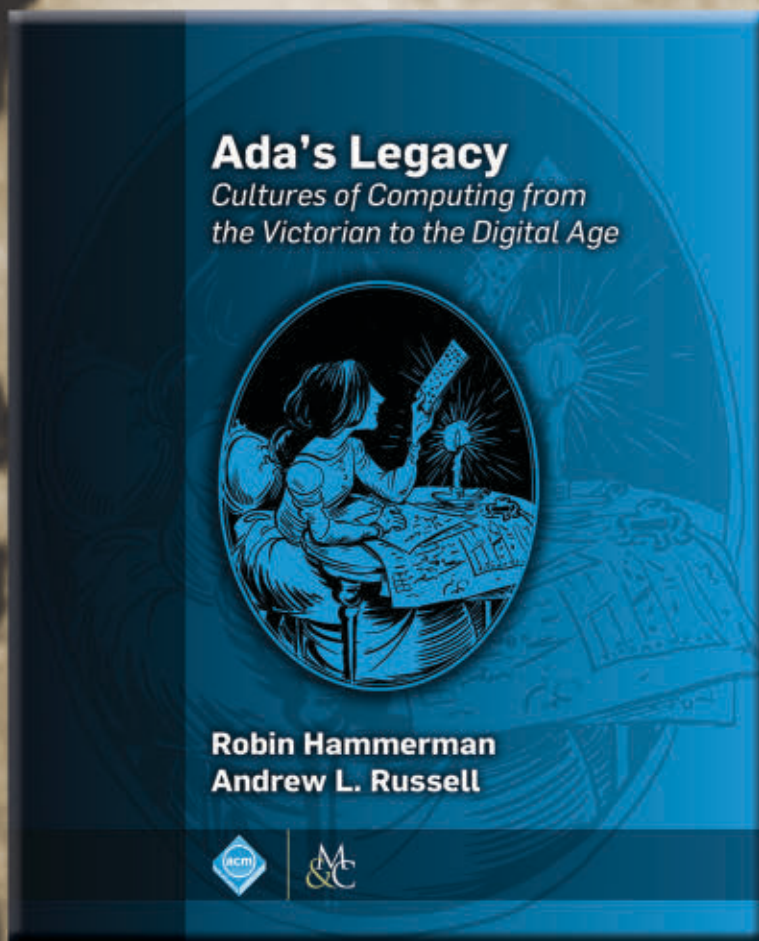
**Enhancing
Symbolic Execution
with Veritesting**

**Plus the latest news
about improving search,
big data blunders,
and digital humanities.**

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>

As the ACM election process begins its biannual schedule, I want to draw attention to the importance of thinking about diversity in our elected bodies. This consideration should be a matter of interest, not only to the nominations committee that I chair this year, but to the electorate as well.

In the past several years, ACM has invested time and effort to become more geographically inclusive through the creation of the ACM-Europe, ACM-China, and ACM-India Councils. We have more work to do along those lines in other regions of geographic importance. Equally important are other axes of diversity including gender and ethnicity. We want our elected officials to reflect the diversity of the computing industry and, frankly, we want to help increase the diversity in this sector. The statistics of involvement in computing are notably out of balance with too few women and too little representation among ethnic groups. The computing community will be well served to increase the participation of diverse constituents and that is equally true of ACM's elected and appointed positions.

I hope you will take these points into account as you consider recommendations for nominees for our elected posts, our awards, and member advancement opportunities. Similarly, as voting begins, please keep in mind the value of diversity on our boards and councils.

Vinton G. Cerf,
PAST PRESIDENT,
CHAIR, NOMINATING
COMMITTEE

Meet the candidates who introduce their plans—and stands—for the Association.

ACM's 2016 General Election

Please take this opportunity to vote.

THE ACM CONSTITUTION provides that our Association hold a general election in the even-numbered years for the positions of President, Vice President, Secretary/Treasurer, and Members-at-Large. Biographical information and statements of the candidates appear on the following pages (candidates' names appear in random order).

In addition to the election of ACM's officers—President, Vice President, Secretary/Treasurer—five Members-at-Large will be elected to serve on ACM Council.

Electronic Balloting Procedures. Please refer to the instructions posted at <https://www.esc-vote.com/acm2016>.

To access the secure voting site, you will need to enter your email address (the email address associated with your ACM member record) and your unique PIN provided by Election Services Co.

Paper Ballots. Should you wish to vote by paper ballot please contact Election Services Co. to request a paper copy of the ballot and follow the postal mail ballot procedures: acmhelp@electionservicescorp.com or +1-866-720-4357.

Postal Mail Ballot Procedures. Please return your ballot in the enclosed envelope, which must be signed by you on the outside in the space provided. The signed ballot envelope may be inserted into a separate envelope for mailing if you prefer this method.

All ballots must be received by **no later than 16:00 UTC on 24 May 2016.**

The ACM Tellers Committee will validate the computerized tabulation of the ballots. Validation by the Tellers Committee will take place at 14:00 UTC on 26 May 2016.

Sincerely,

Gerald Segal

CHAIR, ACM ELECTIONS COMMITTEE

candidates for

PRESIDENT

(7/1/16–6/30/18)

**VICKI L. HANSON**

Distinguished Professor of Computing
Rochester Institute of Technology
Rochester, NY
U.S.
Professor and Chair of Inclusive Technologies
University of Dundee
Dundee, Scotland
U.K.

Biography

Vicki Hanson is a Distinguished Professor of Computing at Rochester Institute of Technology, U.S. (since 2013), Professor and Chair of Inclusive Technologies, Computing, University of Dundee, U.K. (since 2009), and an IBM Research Staff Member Emeritus (since 2009). Previously, she was Research Staff Member and Manager, IBM T.J. Watson Research Center (1986–2008), Research Associate, Haskins Laboratories, New Haven, CT (1980–86), and Postdoctoral Fellow at the Salk Institute for Biological Studies (1978–80).

Vicki is the ACM Vice President. She also currently serves as a member of the ACM Executive Committee and Council, on the ACM-W Europe Executive Committee, and on the ACM Fellows Awards Committee (Chair, 2015). She is Vice President at Large of ACM SIGCHI and an ACM Distinguished Speaker. She has served on the SIG Governing Board Executive Committee (2005–14; SGB Chair 2010–12), and as Chair of SIGACCESS, where she revitalized the SIG and established a successful annual conference (ASSETS). She co-founded the field's premier archival journal (*ACM Transactions on Accessible Computing*) and served as Associate Editor of *ACM TWEB*. She was on the organizing

committee for several SIGPLAN OOPSLA conferences, chaired the recent ACM CEO Search Committee, and currently serves on the Royal Society of Edinburgh Fellows committee (since 2013; Convener 2015).

She is an ACM Fellow, a Chartered Fellow of the British Computer Society, a Fellow of the Royal Society of Edinburgh, and a Senior Member of IEEE. Vicki has served on research advisory boards in the U.S. and the U.K. including the committee that established the U.S. National Science Foundation (NSF) research agenda in Human-Centered Computing. Her research focuses on accessibility of technology for people with disabilities, the aging population, and related research ethics. She has received substantial support for this research from Research Councils U.K., industry, the Royal Society, the Leverhulme Trust, and NSF. She has been honored with a Royal Society Research Merit Award, the Anita Borg Woman of Vision Award for Social Impact (2013), the ACM SIGCHI Social Impact Award (2008), and an IBM Corporate Award for Contributions to Accessibility (2009).

Vicki received her M.A. and Ph.D. from the University of Oregon (Cognitive Psychology, 1976; 1978), receiving the University's Alumni Fellows Award in 1995.

Statement

I am honored to have been nominated for the position of ACM President. ACM is a strong organization, but there are ongoing challenges we must address. With my experience in industry, academia, and within ACM itself, I believe I am well positioned to lead our work on these challenges. I seek your support and your help in making ACM the strongest organization it can be.

ACM is a volunteer-driven society. Throughout my career I have been involved with ACM as a member and volunteer, as an officer at both SIG and SIG Board levels, as a journal Editor-in-Chief, as a member of the ACM Executive Committee and Council, and now as Vice President. In all these roles I have come to admire the dedication of the many people who further our shared community objectives. As a result of their efforts, ACM has the largest membership in its history, a record number of conferences and publications, and a world-leading Digital Library. Our global members are well served through our SIGs and our contributions to a broad range of educational programs for both students and professionals.

Although strong, we must continue to evolve in response to societal shifts, and we must do this without sacrificing our reputation for quality or our financial viability. We must create mechanisms to rapidly identify and support emerging technical areas. We also must continue to find ways to provide value to practitioners. This past year saw new offerings with the launch of the *acmqueue* app

and the Applicative conference, but additional practitioner-focused opportunities are needed. Efforts to interconnect our global computing community as well as our diversity efforts related to women, ethnicity, and accessibility have been well received, but work remains to integrate these into the core fabric of ACM.

In addition to these focus areas, two topics will dominate ACM deliberations going forward: membership and publishing. Both are key to our future.

For membership, ACM must better meet the needs of young professionals. Coming of age surrounded by readily available content, and having increasingly fluid careers shaped by social networking, many wonder what additional value can be derived from ACM membership. As Vice President, I have brought these concerns to the foreground and have started efforts to reach out, such as spearheading ACM's first participation at South by Southwest Interactive 2016 and proposing the formation of a Young Professionals Advisory Board.

In publishing, we must continue to explore how best to address requirements for increasingly open access and the inclusion of rich media and open data in our Digital Library. As we move forward, we must keep in mind that publishing is key to ACM's financial health. Our goal, therefore, must be a plan that financially sustains ACM while addressing these evolving needs.

There is much to do at ACM. With your help and support, as President I will work to ensure ACM initiatives meet the needs of our community.

candidates for

PRESIDENT

(7/1/16–6/30/18)

**ERIK R. ALTMAN**

Director of Technology
IBM Watson Research Center
Yorktown Heights, N.Y.
U.S.

Biography

Erik Altman has been an ACM volunteer for more than two decades and currently serves as ACM Secretary-Treasurer. In addition to general ACM governance, he helps ensure ACM's budget is in good order. (It is, thanks to Darren Ramdin and others at ACM HQ). He is past Chair of the ACM SIG Governing Board (SGB) and its 37 constituent Special Interest Groups. One of his initiatives as SGB Chair was to create the 11 knowledge groups to ease finding SIGs of interest (acm.org/special-interest-groups) as well as chartering SIGLOG. He worked with ACM HQ to pilot daily newsfeeds for each SIG, an incomplete effort on which he would like to renew efforts if elected.

Altman previously served as Awards Chair on the SGB Executive Committee and is immediate past chair of ACM SIGMICRO. Under his watch, SIGMICRO established a Hall of Fame and an Oral History project and relaunched its newsletter. He joined ACM in 1992, and has been a member of SIGARCH, SIGPLAN, SIGSOFT, SIGHPC, SIGMOBILE, SIGMETRICS, and SIGGRAPH. His first paper was in the 1993 Genetic Algorithms conference.

At IBM, Altman helps develop material for monthly readouts to CEO Ginni Rometty and other

senior IBM executives on key technology issues. Earlier at IBM Research he directed two projects: WAIT: on performance tooling; and Liquid Metal—Programming FPGAs and GPUs in a Java-like language. He has co-authored more than 45 papers, and more than 30 patents and pending applications. He was a co-founder of IBM's DAISY binary translation project and an architect of the Cell processor in Sony's Playstation 3. He worked as a hardware and software engineer at Bauer Associates, Machine Vision, and Tek Microsystems. More info: researcher.ibm.com/person/us-ealtman

Other Service: Editor-in-Chief, *IEEE Micro*; Program Co-Chair and General Chair: PACT and NPC Conferences; Program Chair: CASES Conference; Co-Founder and three-year General Chair: Workshop on Binary Translation; Co-Founder: FastPath Workshop and IBM P=ac2 Conference; Dagstuhl Organizer; Guest Editor: *IEEE Computer, Journal of ILP*; *IBM Journal of R&D*; Chair, ACM Student Research Competition at PACT, Keynotes: 2009 PEPSMA Workshop; 2011 CGO Conference, 2013 Cosmic Workshop.

Altman received an S.B. (MIT, 1985), an M.S. (McGill, 1991), and Ph.D. (EE, McGill, 1995).

Statement

"We must think and act anew."

— A. Lincoln

ACM's stated mission is "Advancing computing as a science and profession." The value of science in ACM is clear with 170 conferences globally, 40+ journals, and a Digital Library (DL) of 400,000+ papers. ACM also helps computing professionals via free e-books on CUDA, user experience, AI, and thousands more. ACM offers Learning Packs on security, cloud computing, SaaS, mobile, etc. The relaunched *acmqueue* has great practical content. ACM is also a driving force in keeping undergraduate curricula up to date.

But despite abundant offerings and 100,000+ members, ACM has challenges. It is not clear ACM's financial model based on publishing revenue can endure. Many are not involved in ACM, e.g., the 10+ million programmers worldwide. These challenges are connected and have many causes:

- Difficulty knowing what ACM material others found valuable and why.
- The Open Access movement and free content:
 - stackoverflow helps with targeted questions; recent academic papers are often found elsewhere.
 - A mismatch between most ACM material and practitioner needs.
 - Much ACM content is papers or instruction to teach a new area—not focused material for a how-to Web search like "efficient SQL join."
 - The DL is currently indexed for researchers—by paper titles, authors, conferences.

How to overcome these challenges? "Nothing will come of nothing."^a I propose a mix of technical and social improvements leveraging ACM's unique material:

- Chunk papers, books, and

more in small pieces like paragraphs for search engines to facilitate targeted how-to search.

► Let users guide other users via "star" ratings, easy posting of comments tagged to "chunks," and noting "Users who looked at X also did Y."

- Enhance ACM's international value by machine translations of papers and other material.
- Simple interfaces for ACM users to improve the translations.
- Confirm improvements by a reader vote.
- Reward highly ranked translators.

Indeed, "A little Madness in the Spring is wholesome even for the King."^b Such experimentation can guide ACM to things valued by the computing community, and provide funds to support ACM and its good works.

Such initiatives can also cross-pollinate research and practice, e.g., tagging chunks of research papers of interest to practitioners; conversely providing insight to researchers about practical problems. This interaction could also help ACM quickly identify and support emerging areas via FAQs, conferences, SIGs, etc.

All of these efforts follow a broader theme: inclusion and computer science for all. More accessible and open offerings foster broader participation: globally and by income, gender, age, etc. Broad participation also means hearing everyone and I welcome comments (altmane@acm.org).

It is the most exciting time in my 20+ years of service to be in computing and ACM. It would be an honor to serve as President and help make things even better. I ask for your vote.

a Shakespeare

b Emily Dickinson

candidates for

VICE PRESIDENT

(7/1/16–6/30/18)

**FABRIZIO GAGLIARDI**

Independent Consultant, Geneva, Switzerland
Polytechnic University of Catalonia, Spain

Biography

Current positions:

► Former ACM Europe President and current EUACM chair

► Distinguished Research Director, Polytechnic University of Catalonia, Spain; Senior advisor of the Barcelona Supercomputing Center, Spain

► Visiting professor, Gran Sasso Scientific Institute, L'Aquila, Italy

► Independent computing consultant, Geneva, Switzerland

Other current positions:

Founding member of the Research Data Alliance Organisational Advisory Board

List of previously held positions:

2011–2015: member of the board of Informatics Europe

2009–2015: founding member and chair of the ACM European Council

2009–2013: Director, Microsoft External Research, for Europe, Middle East, and Africa

2010–2012: Principal initiator and chair of the management board of the EU FP7 Cloud Computing Project VENUS-C

2005–2008: Microsoft, Director for Technical Computing for Europe, Middle East, Africa and Latin America

2004–2005: Principal initiator and director of the EU funded project EGEE: Enabling Grid for E-science

2001–2004: Principal initiator and project leader of the EU DataGrid project

2000–2002: Director of the CERN Summer School of Computing

1999–2013: Co-founder of the Global Grid Forum

1985–2000: Several technical and managerial positions at CERN and associated international laboratories

1983–1984: Visiting scientist at the Stanford Linear Accelerator Laboratory of the University of Stanford, California

1978–1982: Principal designer of several on-line data acquisition systems for physics experiments at CERN

1975–1977: Scientific Fellow at CERN

Awards:

2013: ACM Presidential Award, for his instrumental role in creating and leading the ACM Europe Council, composed by some of the most distinguished computer scientists in Europe

2008: Best speaker award at the GridKa computing school in Karlsruhe, Germany (three years in a row)

1999: Price for best track speaker and best overall speaker at the EuroStorage international conference in Berlin.

Statement

I am honored to have been asked to run for ACM Vice President. ACM is a strong organization. Some of that strength comes from recent efforts to be a truly international society. While much has been accomplished with ACM's international initiatives (especially in Europe, India, and China), there is much more to do. I believe my experience in building ACM Europe, combined with the experience gained from being an active member of the European research community throughout my career, gives me the background needed to strengthen and grow ACM's international agenda.

Over the past seven years, I have been responsible for the launch and development of ACM Europe. I helped build the ACM Europe Council and, as chair, drove it to set an agenda for ACM in Europe. As a result of this effort, we have more chapters and members in Europe than at any point in ACM's history. We have engaged with multiple organizations, particularly Informatics Europe, on important issues like computing/informatics education at the secondary level. We issued a joint report on issues in European informatics education and are co-leading and co-funding a two-year research project to really assess the state of pre-university informatics education throughout Europe.

In addition to our efforts on education, we created ACM-W Europe to bring the agenda, resources, and vision of ACM-W to the European issue of women in computing. Several special events have been held, and ACM Europe is gaining traction on this important issue. Finally, I led the effort to create EUACM—ACM Europe's policy arm. EUACM is modeled after USACM and is bringing ACM into important conversations and positioning ACM Europe as a privileged partner of the European Commission funding activity for computer science research and education in Europe.

While we are off to a solid start, there is much more to do—not just in Europe, but throughout the world. My experience at CERN, Microsoft, and now the Barcelona Supercomputer Center, has given me a sense of how an international society should engage with the global computing community. I believe ACM can and should do more—both to build the initiatives we have started and to launch new efforts, particularly in Latin America.

There is much to do at ACM. With your help and support, I believe we can bring ACM to a new level of presence and prominence in the international computing community.

candidates for

VICE PRESIDENT

(7/1/16–6/30/18)

**CHERRI M. PANCAKE**

Professor Emeritus and Intel Faculty Fellow
School of Electrical Engineering & Computer Science,
Oregon State University
Corvallis, OR
U.S.

Biography

Cherri Pancake is Professor Emeritus of Oregon State University and Director of the Northwest Alliance for Computational Science and Engineering, an interdisciplinary research center known for software systems that analyze scientific and climate data to provide information that “makes sense” to decision makers who are not scientists themselves. She is a Fellow of ACM and IEEE.

Early in her career, Pancake conducted ethnographic fieldwork in Guatemalan Indian communities, where she applied cross-cultural techniques to study social change. After earning a Ph.D. in Computer Engineering, she was among the first worldwide to apply ethnographic techniques to identify software usability problems—an approach that is now mainstream—and she conducted much of the seminal work identifying how the needs of scientists differ from computer science and business users.

More recently, she has focused on how “virtual collaborations”—interactions that span large interdisciplinary and physically distributed communities—differ from situations where colleagues

have the opportunity to meet and work together physically. She develops processes and software tools to make remote collaboration and data sharing fit naturally into normal patterns of scientific research and practice.

A member of ACM since 1982, Pancake has been ACM Awards Co-Chair since 2012 and a member of ACM Council since 2013. Over the past 25 years, she has served in a wide variety of roles. Pancake was an area editor for *Communications of the ACM*, led two ACM/IBM industry advisory boards, and chaired the ACM Fellows and Gordon Bell Prize committees. She has held leadership roles with the SC Conference (formerly Supercomputing) since 1990, serving in more roles and over a longer period of time than any other person.

Pancake led efforts to create a new SIG focusing on High Performance Computing—formalized in 2012 as SIGHPC—and has served as Chair since its inception. Under her leadership, SIGHPC grew to over 1,000 members and achieved financial viability in the first year, setting two new records for ACM.

Statement

In more than two decades of ACM volunteer service, I have led major ACM conferences, formed a new SIG, served in a variety of editorial positions, chaired the ACM Awards Committee, and served on ACM Council. I have also held leadership and advisory roles in a number of major national/international research- and data-sharing collaborations, including the Protein Databank, the Long-Term Ecological Research Program, the National Biological Information Infrastructure, and the Network for Earthquake Engineering Simulation. I believe my varied experience, coupled with my unique professional background, has prepared me well for serving you as Vice President.

More than half of ACM’s 100,000+ members live outside the United States. This presents many new opportunities for collaborating with colleagues worldwide, but to take full advantage of them we must first overcome the challenges of distance and culture. While these challenges are not unique to ACM, we do have a unique resource to leverage in overcoming them: our volunteers. I believe the most effective way to overcome the challenges of distance and culture is to actively engage our worldwide membership in shaping the ACM

of the future. Participating in ACM volunteer activities, especially at the national and international level, has broadened my own perspective immeasurably. It is something I think many more people can—and should—be able to experience. As Vice President, I will draw upon my research and practical experience with distributed collaborations to make volunteer opportunities more visible, accessible, and rewarding for all members.

I also believe ACM should be doing more to lead the search for solutions to the “crisis” in business and industry caused by the lack of computing professionals. I recently negotiated with Intel to donate \$1.5 million over the next five years for SIGHPC/Intel Computational & Data Science Fellowships, which explicitly target the need to tap new sources of trained professionals by increasing diversity in the field. If elected Vice President, I will work to establish ACM’s role in this arena. As the leading professional organization for computing, we are uniquely positioned to bring together key stakeholders from academia, industry, and government to develop collaborative models and mechanisms for expanding participation in our field.

candidates for

SECRETARY/TREASURER

(7/1/16–6/30/18)

**CRAIG PARTRIDGE**

Chief Scientist
Raytheon BBN Technologies
Cambridge, MA
U.S.

Biography

Craig Partridge is a Chief Scientist at BBN Technologies, where he has worked on research problems in data communications networks, especially the Internet, since 1983. His contributions include architecting how today's email is routed, co-inventing anycast addressing, and developing the first multigigabit IP router and IP security gateway. Partridge holds 15 patents and has published approximately 60 papers in refereed journals and conferences. He has held adjunct professorships at Stanford University, the University of California at Berkeley, and the University of Michigan.

Partridge has been an active volunteer in ACM since 1987, when he became editor-in-chief of *ACM Computer Communication Review*. A former chair of ACM SIGCOMM, he is a Fellow of the ACM and the current chair of the ACM Fellows Committee. He has served as general chair of ACM SIGCOMM and program co-chair for ACM SIGCOMM and the Internet Measurement Conference. He has been on the program committee for many ACM conferences including the ACM Conference on

Information-Centric Networking, the ACM Internet Measurement Conference, ACM HOTNETS, ACM SIGCOMM, and ACM MobiHoc.

Partridge is also a Fellow of the IEEE. He is a former editor-in-chief of *IEEE Network Magazine*, and a former member of the IEEE Communications Society's Fellows committee and the IEEE Communications Society's Awards committee.

He has served on multiple committees of the U.S. National Research Council, including chairing the committee regarding the behavior of the Internet on 9/11/2001. An early member of the Internet Engineering Task Force, Partridge was a founding member of the Internet Engineering Steering Group (IESG).

His research focus has been on data communications, broadly construed. Any problem that involves getting information from one place to another is of interest to him.

He received his A.B. degree in History, his M.Sc. in Computer Science and his Ph.D. in Computer Science all from Harvard University.

Statement

I have been a member of ACM for 35 years. I am honored to be considered as a candidate for Secretary/Treasurer of ACM.

ACM Secretary/Treasurer has two major roles. The first is overseeing ACM's finances. Currently ACM's finances are good and trends suggest finances will remain good for the next few years. Much of the Secretary/Treasurer's role, therefore, is likely to be ensuring ACM is using its funds to serve ACM members and achieve ACM goals.

That leads to the second role of Secretary/Treasurer, which is to serve as a member of the ACM Executive Committee and participate in decisions that seek to better serve members and better achieve those ACM goals. In that light, I think it is useful to explain my perspectives as an ACM volunteer.

I am deeply interested in ACM's role in exposing the best ideas of practitioners and researchers, through conferences, tutorials,

webinars, publications, and the like. Two examples hint at how challenging that role can be. Recently, we learned that graduate students at ACM SIGCOMM were having trouble understanding the papers being presented because they lacked the necessary background. Our field has become so diverse it is hard for a person to track all the advances, and graduate students are just getting started. So we created a program of short lectures at the conference to provide the background. Those short lectures were so popular we ran out of meeting space. Another example is that about 15 years ago, SIGCOMM decided to put conference papers online, for free, before the conferences took place. There was a fear this would undermine attendance. Instead, conference attendance grew. The moral I take from these experiences is that we have to regularly improve how we make good ideas accessible to our membership.

candidates for

SECRETARY/TREASURER

(7/1/16–6/30/18)

**ELIZABETH CHURCHILL**

Director of User Experience

Google

Mountain View, CA

U.S.

Biography

Elizabeth Churchill is a Director of User Experience at Google. Her field of study is Human-Computer Interaction, and her current focus is on the design and development of connected devices and of developer tools for device ecosystems.

Churchill has built research groups and led research in a number of well-known companies, including as Director of Human-Computer Interaction at eBay Research Labs in San Jose, as a Principal Research Scientist and Research Manager at Yahoo! in Santa Clara, CA, and as a Senior Scientist at PARC and before that at FXPAL, Fuji Xerox's Research lab in Silicon Valley.

Churchill served on the Executive Committee of the ACM's Special Interest Group on Computer-Human Interaction (SIGCHI), for eight years, six of those years as Executive Vice President. She has also held committee positions on a number of ACM SIGCHI-associated conferences.

A Distinguished Scientist and Distinguished Speaker of the ACM and a member of the SIGCHI Academy, Churchill has worked in a number of research areas and has been successful at publishing, prototyping, and patenting. She has more than 50 patents granted or pending, and over 100 publications in theoretical and applied psychology, cognitive science, human-computer interaction, mobile and ubiquitous computing, computer-mediated communication, and social media.

Churchill earned her B.Sc. in Experimental Psychology and her M.Sc. in Knowledge-Based Systems from the University of Sussex in the U.K., and her Ph.D. in Cognitive Science from the University of Cambridge, also in the U.K. Her dissertation research focused on the design and development of Programmable User Models. After her Ph.D., she was a Postdoctoral Research Fellow at the University of Nottingham before leaving the U.K. and moving to industry in 1997.

Statement

ACM is the premier professional organization for computer science as a field of enquiry and application. ACM provides a platform for the best academic research and a bedrock for academics and practitioners alike to engage in life-long learning.

ACM also faces challenges. Membership may be steady, but some view the organization as not addressing the needs of practitioners and not addressing the needs of young people starting their careers—people who are served just as well by informal social networks for career growth. Further, discussions about open content publishing are seen to be at odds with one of the cornerstones of the ACM's business model: charging for access to content.

One of the challenges I would like to address with others in the leadership group of ACM is how we can ensure future growth by providing a better platform for those in early-stage careers

and how we can more effectively capitalize on our global presence, acknowledging academics and practitioners alike. It is my belief ACM can continue to provide the strongest foundational social network for a life-long career in all aspects of computer science.

I have been a member of the ACM for two decades. Volunteer activities to date have been within SIGCHI, one of the largest SIGs. In addition to technical program committee activities, I have chaired conferences and was on the Executive Committee for SIGCHI for eight years, two years as VP for Chapters, and a further six years as Executive Vice President. Key initiatives I worked on with other committee members in those positions were accessibility, HCI and Computer Science Education, and HCI and Policy.

I am honored to be nominated for the position of Secretary/Treasure of the ACM and look forward to continuing my volunteer activities for ACM.

candidates for

MEMBERS AT LARGE (VOTE FOR FIVE)

(7/1/16–6/30/20)

**SUSAN DUMAIS**

Distinguished Scientist, Deputy Managing Director
Microsoft Research
Redmond, WA
U.S.

Biography

Susan Dumais is a Distinguished Scientist and Deputy Managing Director of the Microsoft Research Lab in Redmond, WA, and an adjunct professor at the University of Washington (1997–present). Before joining Microsoft, she was a Member of Technical Staff at Bell Labs and Bellcore (1979–1997). She earned her Ph.D. in cognitive psychology from Indiana University (1979) and B.A. in mathematics and psychology from Bates College (1975).

Her research spans a variety of topics at the intersection of information retrieval and human-computer interaction with a focus on algorithms and interfaces to help people more easily find relevant information and derive insights from it. She is a co-inventor of Latent Semantic Analysis, a well-known dimension-reduction technique for concept-based retrieval. She has developed systems and methods for email spam filtering, desktop search and navigation, context-aware Web search systems, and dynamic information environments. She has also worked closely with Microsoft teams on search-related innovations for the desktop, enterprise, and Web. Her interdisciplinary research on search algorithms and interfaces has been widely cited (more than 45,000 citations), and she holds more than 40 patents.

Dumais is Past-Chair of ACM SIGIR (1999–2003), served as an editor for ACM ToIS (1999–2011) and ToCHI (2001–2011), was technical co-chair of ACM CHI (1994) and ACM SIGIR (2006), and served on several ACM committees including Fellows, Athena Lecturer, and Nominations. She was elected to the CHI Academy (2005), ACM Fellow (2006), National Academy of Engineering (2011), American Academy of Arts and Sciences (2015), and received the ACM SIGIR Gerard Salton Award for Lifetime Achievement (2009), Tony Kent Strix Award (2014), and ACM-W Athena Lecturer Award (2014).

Statement

I have been an active member of ACM for my entire career and am honored by this nomination and the opportunity to serve the community in a new capacity. ACM is widely recognized as the premier professional computer society, with strong commitment from volunteers, and a broad set of activities and resources for members, yet there are also challenges ahead.

ACM has a long history of advancing computing research through its conferences and publications and increasingly through resources for education and professional development. The Digital Library will have to continue to evolve to address the need for open access and the inclusion of rich media, open data, and other resources for students, practitioners and researchers. There are also opportunities to augment conferences with richer real-time engagement and a more permanent record of talks, workshops, and panels.

Many of the challenges the computer science community faces today will not be solved by a single technical discipline in isolation and would benefit from cross-disciplinary perspectives. I would like to see us develop opportunities for people with different backgrounds to come together to address important technical and societal problems. I will also work to continue to broaden participation from individuals with diverse backgrounds and to support them in career development. Finally, I would like to see ACM take a more proactive role in informing policy around important societal issues such as personal privacy, and network and data security.

I believe my sustained service to ACM and multidisciplinary background provide me with perspectives that would be an asset to ACM in addressing computing challenges moving forward. If elected, I would be honored to serve as a Member at Large on the ACM Council.

**EUGENE H. SPAFFORD**

Professor and Executive Director
Purdue University CERIAS
West Lafayette, IN
U.S.

Biography

Eugene H. Spafford (Gene, Spaf) received a B.S. in both math and computer science from SUNY at Brockport ('79). He received his M.S. ('81) and Ph.D. ('86) in CS from Georgia Tech. After a post-doc at Georgia Tech he joined the faculty at Purdue University in 1987 as a professor in CS. He also has courtesy appointments in ECE, Philosophy, Political Science, and Communication.

His current research interests are prevention, detection, and remediation of information system failures and misuse, with an emphasis on applied information security. He also has been deeply involved in issues of science and national security policy. He founded the COAST Laboratory at Purdue in 1992, which became the Center for Education and Research in Information Security and Assurance (CERIAS) in 1998.

Spaf has received many awards for research, teaching, and service, including: National Computing Systems Security Award, ACM SIGSAC Outstanding Contribution, ACM SIGSAC Making a Difference Award, ISSA Hall of Fame, SANS Lifetime Achievement Award, Computer Society Taylor Booth Award, Upsilon Pi Epsilon ABACUS Award, CRA Distinguished Service Award, ACM President's Award, and an Air Force Civilian Meritorious Service Medal. He is a Fellow of the ACM, IEEE, AAAS, (ISC)2, and a Distinguished Fellow of the ISSA.

A member of ACM for over 30 years, he has been a member of or chaired nine major ACM committees including the ACM/IEEE Joint Curriculum Taskforce (1989–1991), and the editorial boards of two ACM journals. He was an ACM representative on the CRA Board of Directors (1998–2007) and has served as chair of the ACM's U.S. Public Policy Council since 1998. He is currently a member of several company, government, and institutional advisory boards, including those of the U.S. Naval Academy, and the GAO.

Statement

I have been actively involved with ACM for over 30 years, as a researcher and as a member involved in issues of security, privacy, ethics, public policy, and education. This has included serving on or chairing several ACM Task Forces and committees, representing ACM as a CRA board member, and serving as chair of the ACM's U.S. Public Policy Council for over a decade. I have been serving as an at-large member of ACM Council for the last four years. Thus, I have broad experience with ACM concerns and activities.

ACM should be more engaged in some of the issues being raised by the increasing use of computing in critical applications. Issues related to computing and networking are now major topics in public life. ACM can play an important role as a trusted source of expertise and information in these matters, and should be involved in discussing them. Whether those discussions are about the role of early computing education, the issues of privacy in a global community, or the role of social media in political dissent, ACM has members and resources that can make a difference. Over the last few decades, ACM has grown into a major international organization, with much more to offer than the presentation and publication of new research results. More than scholars, we can be leaders at the forefront of changes in the world around us, and advocates for appropriate behavior.

As a member of the Council I will continue to support opportunities to help grow ACM as an internationally prominent organization sought out for our technical expertise—not as advocates but as temperate professionals. This will be in addition to strong support of our base functions—conferences, education, publications, the Digital Library, and SIG activities—and of our increasingly global membership.

candidates for

MEMBERS AT LARGE (VOTE FOR FIVE)

(7/1/16–6/30/20)

**GABRIELE ANDERST-KOTSIS**

Professor and head of the Department of Telecooperation
Johannes Kepler University Linz
Linz, Austria

Biography

Gabriele Anderst-Kotsis received her master's degree ('91, honored with the Award of the Austrian Computer Society) and her Ph.D. ('95, honored with the prestigious Heinz Zemanek Ph.D. award) from the University of Vienna. After visiting professor positions at business schools in Vienna and Copenhagen, she accepted a position as full professor in computer science at Johannes Kepler University Linz where she has been since 2002. She is the head of the Department of Telecooperation with a research focus in mobile computing, performance management as well as cooperative and collaborative systems.

Anderst-Kotsis is author of over 150 scientific publications. She has successfully led several research projects, such as the EU-funded networks of Excellence CRUISE and EuroNGI. She is also well known for her active engagement in organizing international conferences, including iiWAS or MoMM.

She is a member of the Austrian Computer Society and was one of the co-founding chairs of the Society's working group for professors in computer science. From 2003 to 2007 she was president of the Austrian Computer Society.

She is a founding member and currently elected Secretary of the ACM Europe Council. In 2014, she was recognized as an ACM Distinguished Scientist.

For Anderst-Kotsis, a pillar in her academic life has been the management of scientific research. She was appointed in 2007 and reelected in 2011 as Vice Rector for Research at JKU Linz. In this role she has been responsible for the strategic planning and support of the University's research agenda.

She is dedicated to the support and promotion of women in ICT, acting as mentor and role model for young female researchers and establishing specific funding and support programs.

Statement

In my scientific career I have always been dedicated to contributing to the larger scientific community, which for me meant being an active member of my national organization, the Austrian Computer Society, and of ACM as the international community for scientists and practitioners in ICT.

Active involvement means for me the individual contributions each of us will make within our own specialized fields as well as initiating and supporting larger collaborative activities. For such activities, ACM has proven to be a valuable and essential platform.

I do strongly believe we can make the most out of any (scientific) community if we do not wait for the benefits and advantages of the community or organization to be delivered to us but rather to focus on what we can contribute and give to the others.

I would be happy to spread and promote this attitude if elected Member at Large and would see it as my major task in the council to bring forward and support initiatives and new ideas for ACM members and ensure ACM receives the appropriate support and infrastructure needed to continue its important work.

**PAMELA SAMUELSON**

Richard M. Sherman Distinguished Professor
of Law and Information
School of Law, University of California, Berkeley
Berkeley, CA
U.S.

Biography

Pamela Samuelson is a 1971 graduate of the University of Hawaii and a 1976 graduate of Yale Law School. After graduation, she worked for the New York law firm Willkie Farr & Gallagher. From 1981 through June 1996 she taught at the University of Pittsburgh Law School. In the mid-1980s, she led the Software Licensing Project of the Software Engineering Institute at Carnegie Mellon University. She joined the Berkeley faculty in 1996. She is now the Richard M. Sherman '74 Distinguished Professor of Law and Information and a Director of the Berkeley Center for Law & Technology.

Samuelson teaches courses on intellectual property, cyber law, and information privacy. She has written and spoken extensively about the challenges new information technologies pose for traditional legal regimes, especially for intellectual property law.

As a Contributing Editor of *Communications of the ACM* since 1990, she has written a regular "Legally Speaking" column. She was named a Fellow of the ACM in 1998. She serves on the USACM Public Policy Committee.

Samuelson is a Fellow of the American Academy of Arts & Sciences, a past Fellow of the John D. & Catherine T. MacArthur Foundation, and an Honorary Professor of the University of Amsterdam. She is President and Chair of the Board of Directors for Authors Alliance, a nonprofit whose mission is to facilitate authorship in the public interest. She has been on the Board of Directors of the Electronic Frontier Foundation since 2000 and has been its Vice Chair since 2009. She is a member of the Advisory Boards for Public Knowledge and the Electronic Privacy Information Center.

Statement

Throughout my career, I have served as a bridge between the fields of law and computing, translating legal developments, such as the copyright "look and feel" lawsuits, to computing professionals, and taking insights from the computing field to inform legal policy debates (for instance, when courts were ruling that mathematical algorithms were not patentable, but non-mathematical algorithms were, I wrote to legal professionals that computing professionals do not think that is a viable distinction).

I have organized debates for ACM conferences about legal issues of concern to computing professionals and written three "Legally Speaking" columns a year for the past 26 years, explaining legal developments of concern so that ACM members could understand the terms of important debates. I have helped formulate some positions that the USACM public policy committee has taken on some of these important issues.

Because legal and policy issues are of growing concern to those in the computing field, what I would bring as a Member at Large to ACM is a wealth of expertise on legal and policy issues, as well as the difficult transitions journals are making in the turbulent field of publishing and the open access options authors often prefer.

candidates for

MEMBERS AT LARGE (VOTE FOR FIVE)

(7/1/16–6/30/20)

**ELIZABETH D. MYNATT**

Professor of Interactive Computing
College of Computing
Georgia Institute of Technology
Atlanta, GA
U.S.

Biography

Elizabeth D. Mynatt is a Professor of Interactive Computing in the College of Computing at Georgia Tech and the Executive Director of Georgia Tech's Institute for People and Technology (IPaT). From 1995–1998, Mynatt was a member of the research staff at the Xerox Palo Alto Research Center (PARC). She earned her Bachelor of Science summa cum laude in Computer Science from North Carolina State University and her Master of Science and Ph.D. in Computer Science from Georgia Tech.

Mynatt is an internationally recognized expert in the areas of ubiquitous computing, personal health informatics, computer-supported collaborative work and human-computer interaction. Named Top Woman Innovator in Technology by *Atlanta Woman* magazine in 2005, Mynatt has created new technologies that support the independence and quality of life of older adults "aging in place," that help people manage chronic disease, and that increase creative collaboration in workplaces. At Georgia Tech, Mynatt played an active role in creating and directing a new Human-Centered Computing (HCC) Ph.D. program, with roots in computing, psychology, cognitive science, sociology, and anthropology.

Mynatt is an ACM Fellow, a member of the ACM SIGCHI Academy, and a Sloan and Kavli Research Fellow. In 2007, Mynatt joined the Computing Community Consortium (CCC) and became vice-chair in 2014. She will chair the CCC through June 2018. As a committee of CRA, the CCC's mission is to "catalyze the computing research community and enable the pursuit of innovative, high-impact research." Mynatt served as general chair of the 2010 ACM CHI conference, ACM UIST'98, and ICAD'97 (International Conference on Auditory Displays). She has served on industry advisory boards, including Microsoft Research's Technical Advisory Board.

Statement

ACM has been at the center of my professional life since my early student years. I have progressed from student to professor, from conference volunteer to conference chair, from mentee to mentor of junior faculty and students. ACM shaped my understanding of computing as a scientific field, profession, and community. ACM is part of my daily existence; writing and reviewing papers, digging into the depths of the Digital Library, planning or attending a conference, and teaching my students how to be informed, productive, and imaginative researchers, educators, and computing professionals. In my academic home, ACM keeps me grounded and connected to our profession across a spectrum of institutions.

I am honored to be nominated as a potential Member at Large. While I have supported a variety of ACM conference and publication activities, I am enthusiastic about serving ACM as a whole on behalf of a tremendous computing community. We face many challenging issues as our field and community rapidly changes to meet new demands and opportunities. Our universities face rapidly increasing enrollments but many of these new students may not identify as computing professionals. Discussions around open access to publications will transform into new challenges around open access to data as structures and expectations for scientific collaboration evolve. Even more challenging are the needs for sustained leadership as computing plays an increasingly important role in society. Challenges abound as our profession grapples with the tight interplay between the social and the technical in domains such as health-care, "smart" cities, transportation, and communication. Much is possible and ACM has the responsibility to engage its membership and society as a whole in charting this future.

**MANINDRA AGRAWAL**

Professor, Department of
Computer Science and Engineering
India Institute of Technology Kanpur
Kanpur, India

Biography

Manindra Agrawal did his B.Tech and Ph.D. in computer science from IIT Kanpur in 1986 and 1991, respectively. His thesis was in complexity theory. After graduating, he worked at Chennai Mathematical Institute and the University of Ulm (Germany) as Humboldt Fellow before joining IIT Kanpur as Assistant Professor in 1996. His areas of research are complexity theory, computational number theory, and algebra.

He is a recipient of several awards and honors including the Gödel Prize, Fulkerson Prize, Humboldt Forschungpreis, and Infosys Prize in Mathematics. He is a Foreign Associate of the National Academy of Sciences (U.S.), and a Fellow of the World Academy of Sciences, Indian National Science Academy, and Indian National Academy of Engineers.

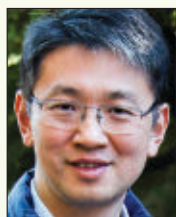
Statement

While ACM has a strong presence in many countries of the world, it has only recently started expanding its presence in India. India has a large number of IT and CS professionals, and ACM can provide them an excellent platform to contribute to the domain as well as learn from others. As a member at large, my aim would be to help in the expansion of ACM activities in India and nearby regions.

candidates for

MEMBERS AT LARGE (VOTE FOR FIVE)

(7/1/16–6/30/20)

**YUNHAO LIU**

Chang Jiang Chair, Professor,
and Dean of the School of Software
Tsinghua University
Beijing, China

Biography

Yunhao Liu is Chang Jiang Chair Professor and Dean of the School of Software at Tsinghua University, China. He is also an ACM Fellow and IEEE Fellow.

Yunhao received his B.S. degree from Tsinghua University's Automation Department in 1995, and M.S. and Ph.D. degrees in Computer Science and Engineering from Michigan State University in 2003 and 2004, respectively.

He joined the Hong Kong University of Science and Technology as an assistant professor in 2004, and was promoted in 2008 to associate professor. Yunhao served as the postgraduate director in the computer science department in HKUST from 2009 to 2011. He joined Tsinghua as Professor and Chair in 2011. He was elected to be the Dean of the School of Software in Tsinghua in 2013.

Yunhao is currently the Chair of the ACM China Council. He served as the Associate Editor-in-Chief for the *IEEE Transactions on Parallel and Distributed Systems* from 2011–2014, and he is now Associate Editor for the *IEEE/ACM Transactions on Networking* and *ACM Transactions on Sensor Network*. Among the awards he received were four Best Paper awards at several leading international conferences, such as ACM MobiCom 2014, as well as many other prestigious honors including the 2012 ACM Presidential Award, the 2011 China National Natural Science Award, the National Distinguished Young Scholar Award, and the Hong Kong Best Innovation and Research Award in 2008.

Statement

I have been an active member of ACM for many years. When ACM started to expand international influence by forming regional councils in Europe, China, and India, I was closely involved with the creation of the ACM China Council and have been active in its operation since 2009. After I became the Chair of ACM China Council in 2013, I became involved in many aspects in promoting the development of ACM in China.

In the past few years, the ACM membership in China has increased from about 1,000 members to more than 15,000 members, and the number of local chapters has increased to more than 20. Professional activities at the national and chapter levels are growing. The ACM China Rising Star Award and ACM China Doctoral Dissertation Award have been endowed for two years.

The rapid development of ACM in China made me realize that while ACM is an internationally recognized computing association, there is still much that could be done in the global arena. For many professionals in China, ACM is still unreachable: they can rarely hear the lectures from Turing Award Laureates, ACM Distinguished Speakers, or even international experts, nor attend ACM's top conferences. To my knowledge, this happens not only in China, but also in many other regions and countries.

Based in the U.S., ACM is a resource that could benefit computer scientists throughout the world, and I would like to work with ACM to grow its activities and create more links within the community of computer scientists, and to see more ACM members benefit from the links and bridges ACM built for them to advance their computing careers and broaden their networks.

**PAUL G. SPIRAKIS**

Professor in Computer Science
University of Liverpool
Liverpool, England
U.K.

Biography

Paul G. Spirakis is Chair and Professor in the Computer Science Department at Liverpool and an international leader on algorithms, complexity, distributed computing and algorithmic game theory. His work is characterized by the use of Applied Probability.

Spirakis (born 1955) got his five-year Engineering Diploma from the National Technical University of Athens (NTUA), his M.Sc. and his Ph.D. from Harvard University. He has been a faculty member of NYU Courant (U.S.) and is also a professor at Patras University (Greece). He directs the initiative of Networks Sciences and Technologies of the School of EEE-CS at the University of Liverpool. He is also the President of the Computer Technology Institute (CTI) of Greece.

He has co-authored over 380 papers in the most significant journals and conference proceedings in computer science. He has co-authored three scientific books and several book chapters.

He is a Member of Academia Europaea and an inaugural Fellow of the European Association for Theoretical Computer Science (EATCS) as well as its Vice-President. He is a Member of the ACM Europe Council, has been the Chair of the EU ERC Panel on Informatics (2015–16), a member of the EU ISTAG, and a National Rep. in the IST Committee of the EU.

Spirakis has led scientific teams in 19 EU projects and has led 10 Greek research projects, and has chaired the program committees of some of the most prestigious conferences in the Foundations of Computer Science. He is the Editor-in-Chief of *Theoretical Computer Science (TCS)* and is an editor for *Algorithmica*, *JPDC*, *Theory of Computing Systems*, *PPL*, *Acta Informatica* and *Computer Science Reviews*.

Statement

I have been an ACM Member for many years and I currently serve as a Member of the ACM Europe Council. I have organized such ACM conferences as STOC and SPAA and have chaired the PCs of ACM SPAA and PODC. I believe I can contribute toward spreading the values of ACM in Europe at large and toward helping ACM to advance the computing profession, especially in countries that critically need it (like Greece).

candidates for

MEMBERS AT LARGE (VOTE FOR FIVE)

(7/1/16–6/30/20)



JUDITH GAL-EZER

Professor of Computer Science

Open University of Israel

Ra'anana, Israel

Biography

Judith Gal-Ezer is a professor of computer science at the Open University of Israel (OUI). She served as Vice President for Academic Affairs at the Open University for almost ten years: 1999–2005, and 2009–2012.

She developed numerous courses in Mathematics and CS, was one of the designers of the CS undergraduate program, and later the CS Masters. She served as head of the Development division, and as head of the Mathematics and Computer Science department. She has served as Advisor to the President on Women's and Gender Issues since 2014.

After several years of research work on wave propagation and seismology, her research turned to teaching computer-integrated mathematics and CS education, which became her main area of interest.

She is the ACM SIGCSE 2007 Special Contribution to Computer Science Education award winner, and the recipient of the IEEE 2015 Computer Society Taylor L. Booth Education Award.

Gal-Ezer co-supervised numerous Ph.D. students and Master students. She serves on professional journals editorial boards and was an associate editor of *ACM Inroads*.

Professional Committees. Gal-Ezer was on the Ministry of Education professional committee which put together the CS curriculum for Israeli high-schools which is considered as a breakthrough in CS education worldwide. She later chaired this committee, and is now again a member of this committee. She also served as a member of the CHE, the Israeli Council for Higher Education.

Gal-Ezer serves on the advisory council of the Computer Science Teacher Association (CSTA, founded by the ACM), and on the ACM-Europe Council and EUACM.

For more information, see http://www.openu.ac.il/Personal_sites/judith-gal-ezerE.html

Statement

My main area of interest is computer science education research. This includes designing curricula for different levels of studies; designing CS teacher training; examining misconceptions of computing concepts, and addressing problems in the teaching of complex concepts. Being engaged in CS education motivated me to examine diversity issues, trying to understand the existing gender gap in schools universities and industry.

If elected, I would strive to place CS on par with other sciences: physics, chemistry, and biology. In countries where sciences are mandatory, CS should be too. If the other sciences are electives, CS should be included among them. CS should be counted for credit wherever sciences are counted for credit.

Increasing the availability of CS education in schools is one of the strategies suggested to address the widely reported gender gap. From my point of view, young people in schools—male and female—should be exposed to the various facets of this discipline. CS curricula should address abstraction and problem solving, writing robust algorithmic solutions, considering their correctness and efficiency, implementing them in a programming language, and running them on a computer.

Most important, teachers qualified to teach CS should have appropriate, formal education. This means teacher training programs are needed where CS is an integral part of the school system.

CS education research tools and methodologies also require being developed for conducting scholarly research.

I hope to join forces with ACM efforts already underway in convincing stakeholders and policy makers toward making real progress in placing computer science on par with the other sciences, and bring women and minority groups into computing.

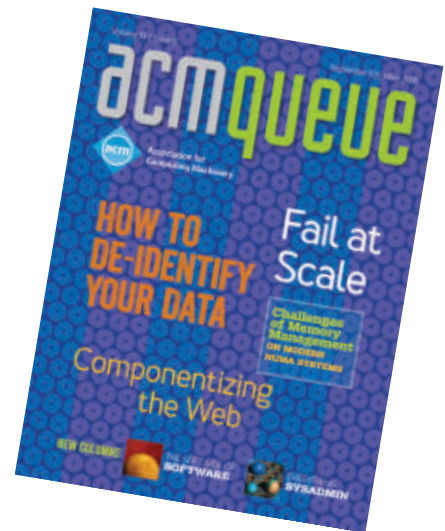
acmqueue

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.

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:10.1145/2898963

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

Bringing Computer Science to U.S. Schools, State by State

Mark Guzdial considers the crucial role of states in Computer Science for All.



Mark Guzdial
**State of the States:
Progress Toward
CS for All**

<http://bit.ly/24EaYfW>

February 24, 2016

On Jan. 30, U.S. President Obama announced the initiative *Computer Science for All* (see announcement at <http://1.usa.gov/1RLcR3s>). The goal is to give every student an opportunity to learn computer science. His proposed budget is over \$4 billion, primarily in the National Science Foundation and Department of Education budgets.

School education in the U.S. is not controlled by the federal government. While the federal government can provide funding, the individual states and territories make decisions about curriculum, teacher certification and professional development, and standards. In the end, only the individual states and territories can realize the goals of *Computer Science for All* (CSforAll).

Coincidentally, the day after the president's announcement, a group of state and territory leaders who belong to the Expanding Computing Education Path-

ways (ECEP) Alliance presented their five-year plans at a meeting near Washington, D.C. Leaders from Alabama, California, Connecticut, Georgia, New Hampshire, South Carolina, Maryland, Massachusetts, Puerto Rico, Texas, and Utah described how they plan to grow CS, broaden participation in computing, and develop teachers. These plans give us insight into the progress toward and challenges to achieving CSforAll.

Growing CS

The ECEP cohort is making strides toward making high school-level CS curricula available in their states and territories: *Exploring Computer Science*, *Computer Science Principles*, and *Advanced Placement CS Level A*, which are being adopted across the U.S. Alabama was the first state to include *CS Principles* in its state curriculum. *Project Lead the Way* (PLTW) is offering professional development and implementing its new computing curricula, but the cost of PLTW may be prohibitive for many schools. Code.org is partnering with selected districts and states, including Alabama and Utah, to provide teacher pro-

fessional development and curricula, and is partnering with groups in several other states (including Massachusetts, South Carolina, and Georgia) to provide services like elementary school teacher professional development. States are also defining their own classes. Georgia has created five new CS classes in its state curriculum. With seed funding from ECEP, Puerto Rico has started an ambitious effort to translate *Exploring CS* into Spanish, and had several teacher leaders attend *Explore CS* professional development in Chicago.

It is going to take time to reach CSforAll. Most of the ECEP cohort expect to offer CS in 30%–50% of all high schools in their state within the next five years. Puerto Rico is just introducing *Exploring CS* into its first schools this year. In several states, the goal is to have a single CS teacher in 50% of schools within five years, which is insufficient to provide access to CS education for all students.

Funding is a significant limitation to growing CS in the states and territories. The federal government funds Career and Technical Education (CTE) classes through the Perkins Act. Differences in interpretation of the Perkins Act can limit funding of CS curricula and professional development. Perkins CTE funds are not available for CS education in states and territories where CS is not considered CTE. Some states and districts are debating who may teach CS (CTE teachers, STEM teachers, and/or teachers with other certifications) and whether Advanced Placement (AP) courses can serve as Perkins-required assessments at end-of-pathways. In some states, STEM

or CTE teachers are given special incentives, but CS is not included in that definition. In Texas, school districts receive additional state funding for students enrolled in CTE classes, but most CS courses (including AP CS A and CS Principles) are not under the CTE umbrella. There is a disincentive for schools and districts to invest in building CS course pathways.

States are developing CS standards, which define learning goals for their students. Massachusetts released its Digital Literacy and Computer Science standards for comment (<http://bit.ly/1Rshtc6>) with the goal of finalizing state approval in June 2016. Having standards makes it easier to define classes, to create teacher certifications, and to grow teacher professional development programs.

CS is an elective subject in much of the U.S. Over half the states and territories allow CS to count toward some high school graduation requirement (such as toward a required number of courses in mathematics, science, or foreign language). Making CS count for high school requirements can be difficult where graduation requirements are decided locally, at the county, district, or school level. California hopes to persuade its university system to value CS in admission decisions, as a way to incentivize student participation in CS classes. Massachusetts is looking to have CS “count” as meeting a mathematics requirement in the recommended MassCORE college and career readiness curricula.

Several states are holding summits to organize around promoting CS education. Maryland has one planned for April 12. These summits have been useful in many states (California, South Carolina, Massachusetts) to create a coherent agenda across legislators, state department of education representatives, higher education, and industry leaders.

Developing Teachers

The greatest barrier to growing CS is growing the number of teachers. Many in the ECEP cohort have CS teacher certifications or are creating them now. These are important to incentivize teachers (because teachers can get a certification for developing CS expertise) and to provide quality CS education (because schools can use certification in selecting teachers). Both Utah and South Carolina are implementing multitiered teacher certification sys-

tems so teachers get higher levels of certification with more professional development. Several states are resisting the creation of yet another teaching credential, with its associated costs.

Making professional development available to teachers is a top priority. Texas is exploring online options to reach its teachers, especially in rural regions. Texas is offering a one-time stipend of \$1,000 for teachers who add a CS certification to their professional portfolio. The Hour of Code from Code.org is playing an important role in reaching students, but also teachers. Many teachers have a perception that CS is difficult and requires extensive, expensive training. Hour of Code activities can help change that. Utah’s efforts to grow CS education started with Carl Lyman introducing the Hour of Code to Utah education department staffers, which changed their perceptions of CS.

After developing CS teachers, the challenge is to retain them. Utah is concerned about losing CS teachers to industry or retirement. Alabama has been pleased with retention rates so far, but is concerned about the future. Overall, there is a shortage of teachers in the U.S., and teachers do not stay in the classroom long. CS teachers may be particularly hard to retain, given the high demand outside the classroom for their computer science expertise.

Broadening Participation

States and territories are seeking strategies to provide computing education to more female, Hispanic/Latino, and Black/African-American students. A first step in many states and territories is simply to create a mechanism to find out who is taking CS in the state. Most education departments/offices track who takes classes in mathematics, science, reading, and other core subjects, but not CS because it has not been a core subject. Several of the ECEP states are changing how they track CS student enrollment. In Connecticut, legislators have just required CS to be part of the K–12 curriculum, and now Connecticut is conducting a survey to measure the current status of CS education. States have problems getting schools to appropriately classify (“code”) CS classes using state and/or national course classifications, so computing courses can be counted and differentiated.

While the goal is *CSforAll*, the reality will be that CS is most likely to be in the more urban and wealthy schools for several years yet. None of the ECEP cohort expects CS to be in 100% of their schools in five years. South Carolina just completed a “landscape survey,” 72% of all CS classes in South Carolina are offered in just the three largest cities, and poorer schools are 21% less likely to have access to computing education. California is now working on its own survey on the landscape of CS education. To reach further, several states are using online classes to provide access to CS education to schools that do not have a CS teacher.

The goal of broadening participation is about access *and* success. To broaden participation in computing, we need female and underrepresented students to take CS classes and *succeed* at them. Several states have programs to increase the success rates of female and underrepresented minority students in classes like Advanced Placement CS, but that only reaches a small number of students.

Whoever sits in the president’s chair in 2017 will not see the goals of *CSforAll* reached, even if the initiative continues to be funded. There is simply too much ground to cover. Within five years, we may have some CS education in 50% of the schools in the leading states and territories.

The *CSforAll* initiative is important to fund. There is bipartisan support to promote CS education, as a recent op-ed from Republican and Democratic governors suggests (<http://bit.ly/1Qvtf5s>). Reaching all students will take significant effort on the part of educators, school administrators, business leaders, and state and federal leaders. There is already momentum in the ECEP cohort. A handful of other states are also well positioned to make progress toward achieving *CSforAll* goals. The announcement of the initiative has already increased awareness of the status of CS education and is moving state, territory, and national leaders to action.

(My thanks to my colleagues Rick Adrion, Sarah Dunton, Barbara Ericson, Renee Fall, and many ECEP cohort leaders for review and contributions to this.)

Mark Guzdial is a professor at the Georgia Institute of Technology.

© 2016 ACM 0001-0782/16/05 \$15.00

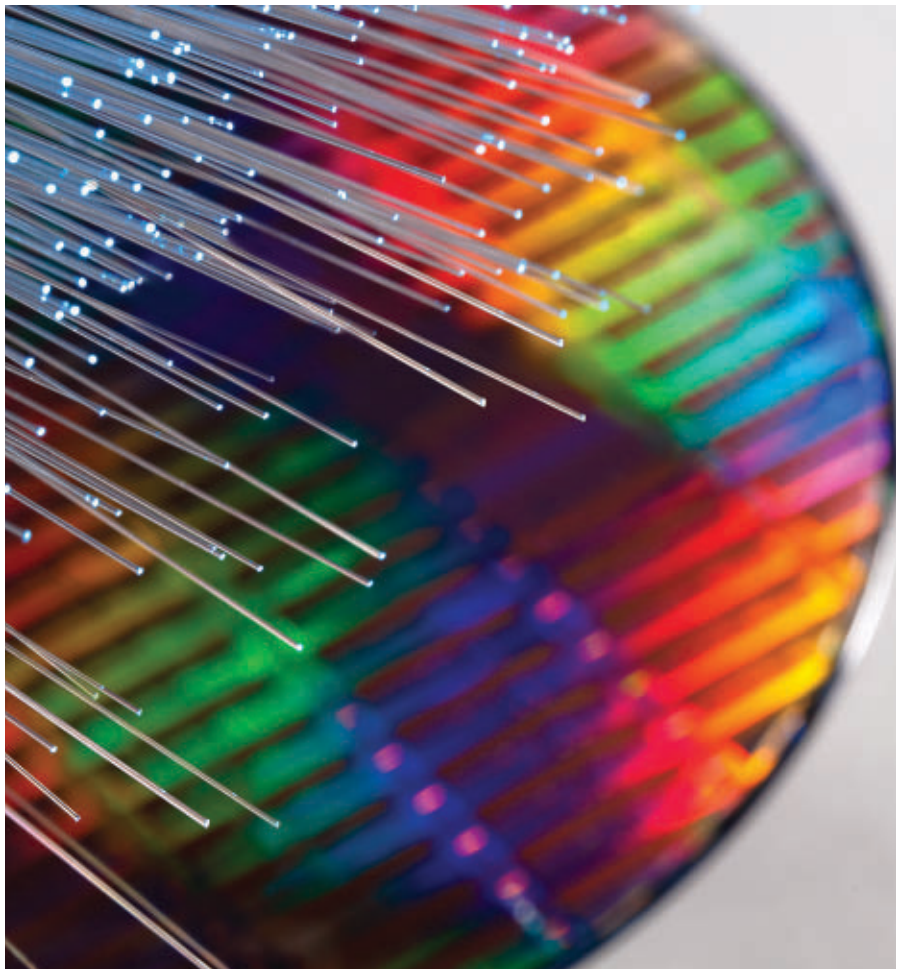
Silicon Photonics: Ready to Go the Distance?

Processes for making CMOS chips are adapted for optical components.

ALTHOUGH PHOTONS OF light dominate the long-distance information superhighways of the Internet, electrons remain the workhorses that move and process information at the local level. Visionaries have imagined for decades that power-efficient light-based communication would migrate to ever-shorter distances and eventually move onto silicon chips alongside transistors at the centimeter scale.

After more than a decade of research and development, chips with onboard photonics are poised to enter the marketplace. Rather than connecting nearby electronic circuits, however, these devices aim to provide high-bandwidth connections over hundreds of meters or longer. Some devices could provide customized solutions for servers and exascale computing, whereas others target the large but cost-sensitive data center market.

To address these applications, technologists have adapted and repurposed the latest processes for fabricating complementary metal oxide semiconductor (CMOS) chips to create optical components. Together with novel packaging schemes to connect chips with optical fibers



Clear curve fibers on a silicon photonics wafer.

IMAGE BY JEFFREY TSENG. COURTESY OF INTEL NEWSROOM

and standards to ensure interoperability, these silicon-photonics chips could at last bring light and silicon into intimate contact. If this framework is commercially successful, it could also open the door for realistic consideration of other ideas for chip-based optics of which researchers have long dreamed.

Putting the Pieces Together

Optical communications systems require a broad palette of components. These include lasers to supply light of various colors (wavelengths), modulators to imprint signals on them, devices to combine and separate wavelengths, waveguides to route light, and detectors to extract the signal. Processes have been devised to build all of these—except the light source—using standard silicon processing techniques.

Probably the easiest components to make are waveguides that precisely route light signals on a chip. These can be made by creating a rectangular wire of semiconducting silicon, which is transparent to infrared light but has a high index of refraction that traps the light, and surrounding it with silicon dioxide insulator. Silicon-on-insulator processes, which are now widely available for CMOS, can easily create a low-loss waveguide by patterning the same semiconductor layer used for transistors and then covering it with insulator.

Another device that has benefited from recent innovations in CMOS processing is the light detector. Alloys of silicon and germanium are now used by many manufacturers, so these materials can now be used without risk to transistors. The alloy composition can be chosen to absorb the signal light that passes freely through silicon, liberating electrons that can be detected electrically.

Because any modifications to the CMOS process pose a risk to transistors, it is difficult to develop a credible optical functionality on chips without access to the process line. William Green, from IBM's Research Center in Yorktown Heights, NY, says he and his colleagues worked closely with engineers at their Burlington, VT, manufacturing facility to implement their optical devices.

The resulting process is now formalized in a kit that lets external clients design optical components, along with analog and digital electronic circuitry.

IBM developed an approach in which mechanical structures are added to completed lasers to let them self-align to the silicon phototonics with sub-micron accuracy.

"It's almost transparent to the designer that there are optical components," Green says. IBM divested its foundry operations to GlobalFoundries in July 2015; Green's team still works with the Burlington team and its clients.

The IBM process includes detectors incorporating pure germanium for operation in the 1.3-micron wavelength range that is specified in interconnection standards. It also supports sophisticated devices that use interference effects to separate slightly different wavelengths of light. Sending multiple signal beams down the same fiber, or wavelength-division multiplexing, allows for a very high bandwidth interconnection with electronics that only has to operate at a fraction of the speed. For example, last year the IBM team showcased their technology with a 100Gb/s transceiver reference design that combined four wavelengths, each carrying 25Gb/s.

Light Sources

For detectors, the intrinsically weak optical coupling of silicon and germanium to light can be overcome using waveguide-based designs. For light sources, however, these group-IV semiconductors are at a serious disadvantage relative to traditional laser materials based on III-V semiconductors such as GaAs, InP, and their relatives.

Direct growth of III-V materials during silicon processing is difficult for a variety of reasons. In one alternative approach, researchers demonstrated laser action from silicon itself under some conditions. However, the

ACM Member News

FOLLOWING THE STRESS OF MULTITASKING



Mary Czerwinski is designing technology that empowers and enriches users' lives. As a

Research Manager in the Visualization and Interaction (VIBE) Research Group at Microsoft headquarters in Redmond, WA, which focuses primarily on topics in human-computer interaction (HCI) and information visualization, Czerwinski's research centers on emotion tracking, task management, multitasking, and awareness systems.

"I am a cognitive psychologist by training. My Ph.D. was in visual attention and multitasking. I stumbled into the field of human-computer interaction in my last year of graduate school," Czerwinski said. "When I started working, it was the perfect time to be a cognitive psychologist; there were no graphics or buttons, users just looked at text and used visual attention to group things. So I went into the field of human-computer interaction to study the ways humans interacted with computers, and I never stopped looking at multitasking."

At Microsoft, Czerwinski is focused on how multitasking creates stress. "The negative health concerns associated with stress—such as the risk of heart disease, obesity, diabetes—led me into affective computing," she noted. "In affective computing, the goal is to have a system, robot, or agent that is emotionally sentient with the user with whom the system is interacting, and a large part of my research is centered on this now."

"While it is easier to detect a user's emotions now," adds Czerwinski, "it is difficult to do it accurately, and the biggest challenge in HCI remains what you do with the information about a user's emotional state."

—John Delaney

most mature strategies use hybrid integration, in which III-V materials are grown in separately optimized systems and then brought together with the silicon.

Intel, for example, has described a hybrid silicon laser in which a stack of III-V materials is bonded to a silicon wafer containing a predefined waveguide. The resulting sandwich is then processed to create a laser with a selected wavelength, whose output is tightly coupled to the waveguide. Intel had promised 100Gb/s silicon photonics chips in 2015, but the planned release was pushed into 2016 because of unspecified production problems. Intel declined to comment for this story.

IBM has developed a different approach, Green says, in which mechanical structures are added to completed lasers to let them self-align to the silicon photonics with sub-micron accuracy. These efforts highlight the critical need for low-cost, precise packaging for optical signals if silicon photonics is to reach its potential.

Zero-Change Process

An alternative to intimately bonding a laser to the silicon chip is to supply its light using a fiber from above, using a structure on the chip to direct it laterally into a waveguide. “We want to treat the lasers like any other power supply in the system,” says Vladimir Stojanović of the University of California, Berkeley. Companies like Luxtera, which is another major player in silicon photonics, have refined active alignment techniques for this sort of packaging, he says.

Stojanović and his collaborators at Berkeley, the University of Colorado, and the Massachusetts Institute of Technology are championing a “zero-change” process that constructs optical devices using an unmodified CMOS flow. In a long-running project supported by the U.S. Defense Advanced Research Projects Agency (DARPA), they are aiming to use optical connections to relieve the information bottleneck between processors and memory. In December, they published a demonstration of a 2.5Gb/s optical link between a full processor and 1MB memory on separate chips, each fabricated alongside the optics using IBM’s SOI foundry process.

This demonstration used only a single wavelength of about 1.18 μ m, which is detected by the silicon-germanium alloy used in the CMOS process. The design, however, could modulate multiple beams in a single waveguide, through a series of micro-ring modulators about 10 microns in diameter, which can be tuned to different wavelengths. The downside of these devices is they are very sensitive to process variations as well as temperature, so the team included an active circuit to lock them to the desired wavelength. This effort was important, Stojanović says, because stability is “the most critical thing that stands in the way of people adopting rings more broadly.”

Indeed, most silicon-photonics projects merge multiple beams after encoding them with data using single-pass devices called Mach-Zender modulators, which are more stable but generally more than order of magnitude larger. For the transceiver market, though, IBM’s Green says the area of the silicon is not a big part of the cost relative to packaging and testing.

On-Chip Optical Memory

One function that could be useful for optical chips is direct storage of optical information. Last year, Harish Bhaskaran of the U.K.’s Oxford University and Wolfram Pernice of the Karlsruhe Institute of Technology and the University of Muenster in Germany, with their colleagues demonstrated a non-volatile memory on a silicon wafer that is written and read optically.

The team exploited an alloy of germanium, antimony, and tellurium, denoted GST, that stably but reversibly switches between a crystalline atomic arrangement and a disordered, amorphous phase. Such phase-change materials have been investigated for electrical memories because the two phases differ in resistivity. Moreover, GST is used in commercial rewriteable optical storage discs, exploiting differences in the reflection of visible light. The new scheme instead exploits the difference in the absorption of light with the 1.5-micron wavelength used for telecommunications.

The team placed a GST film on top of a rectangular waveguide on a wafer, close enough to suck out light energy traveling along the waveguide.

A high-power pulse that melts the GST, when followed by fast cooling, produces an amorphous film. Gentler heating returns the GST to the crystalline state. In the demonstration device, the larger absorption of the crystalline state reduces the light transmitted down the waveguide by about 20%, a difference the researchers can easily detect in the lab. In fact, they say their measurement can discriminate many degrees of crystallization, allowing the storage of multiple bits, because the absorption level drifts less over time than the corresponding resistivity, even when the reading beam is present. Multiple bits would allow more compact optical storage, although probably still much larger than electronic memory cells. Bhaskaran suggests the scheme could help relieve the communication bottleneck and might be particularly useful in future designs based on neuromorphic principles.

Although the kind of on-chip communication these memories might enable could be important someday, for now “the need is really pressing on chip to chip,” Stojanović says. If on-chip optics becomes widespread for inter-chip applications, “we could make things easier for the optics to conquer that last millimeter.” 

Further Reading

Bainbridge, L.
Ironies of automation. New Technology and Human Error, J. Rasmussen, K. Duncan, J. Leplat (Eds.). Wiley, Chichester, U.K., 1987, 271–283.

Monroe, D.,
Still seeking the optical transistor,
Communications, October 2014.

Monroe, D.,
Neuromorphic Computing Gets Ready for the (Really) Big Time, *Communications*, June 2014

D.M. Gill et al.,
Demonstration of Error Free Operation Up To 32 Gb/s From a CMOS Integrated Monolithic Nano-Photonic Transmitter, *IEEE J. Quant. Elect.* 21

C. Sun et al.,
Single-chip microprocessor communicating directly using light, *Nature* 528

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

© 2016 ACM 0001-0782/16/05 \$15.00

Cybersecurity Gets Smart

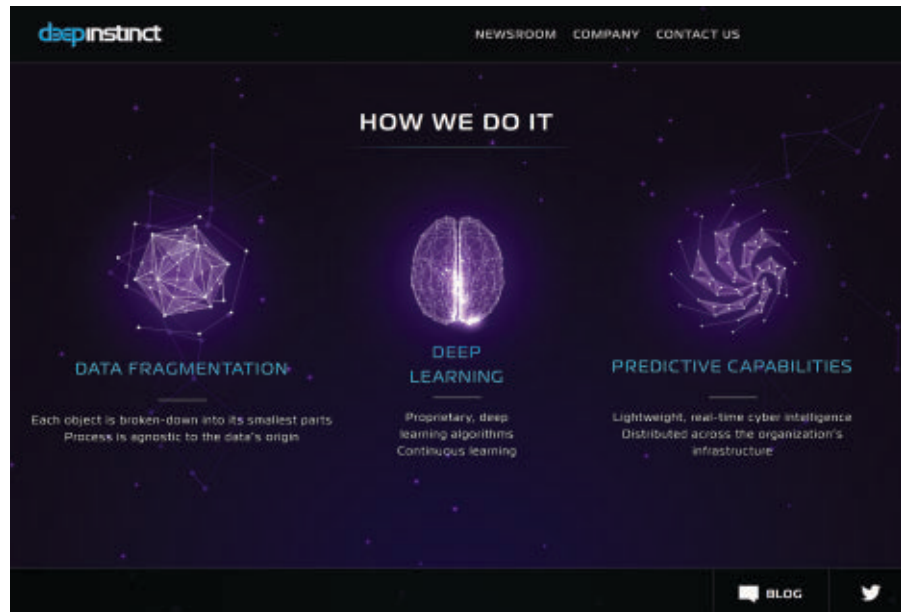
Researchers aim to apply artificial intelligence and machine-learning methods to take cybersecurity to a new, higher, and better level.

OVER THE LAST decade, cybersecurity has careened into the danger zone. Corporations, governments and individuals increasingly have found themselves staring down the barrel of hacks, attacks, and breaches. What is more, as the intensity and frequency of these incidents grows—and far more sophisticated malware, social engineering, and state-sponsored cyberterrorism takes shape—researchers and other security experts are desperately searching for new and better ways to address threats. The traditional approach of using signature-based malware detection, heuristics, and tools such as firewalls and data loss prevention (DLP) simply is not getting the job done.

“We have entered a different era. As society digitizes everything of value, we create irresistible targets for people who want to engage in criminal activities,” observes Bruce Daley, principal analyst at technology research and consulting firm Tractica and author of *Where Data Is Wealth* (Play Technologies, 2015). Today, Daley says, “We’re seeing a level of ingenuity and sophistication from criminals that places even the most modern and sophisticated IT systems at risk.”

Adds Duen Horng Chau, an assistant professor in the College of Computing at the Georgia Institute of Technology, “Traditional security methods aren’t keeping up with cyberthieves. Organizations are constantly forced to react and respond to attacks.”

As a result, researchers are now developing systems that tap artificial intelligence (AI) and machine learning to create radically different, far more sophisticated defense models. These techniques revolve around technologies such as big data, pattern mapping and matching, cognitive computing,



Israel's Deep Instinct describes how its security software utilizes an artificial neural network.

and deep learning methods that simulate the way the human mind works. The goal, quite simply, is to better identify suspicious patterns and behavior, and build security frameworks that are more resilient and adaptable—often on their own. Says Chau, “Malware and threats have been around since the invention of computers, and they aren’t going away. But a variety of new techniques offer hope that we can take a more effective approach.”

Protection Schemes

The scope of today’s cybersecurity problem is somewhat mind-boggling. Security breaches due to malware cost victims more than \$500 billion each year worldwide, according to a 2014 study by market research firm IDC and the University of Singapore.

The problem is growing worse by the day. Ponemon Institute’s 2015 Cyber-Crime Study found that the cost of digital crime rose by 19% in the last year,

and the average annual loss to companies worldwide is \$7.7 million. “The number of files that a security company must deal with now exceeds hundreds of thousands every day and it’s continuing to grow,” Chau says. “Manual approaches and signature-based approaches are no longer effective.”

Adding to the problem is the growing prevalence of zero-day attacks (those that strike and spread immediately), polymorphous malware (which can alter its code to escape detection), viruses, Trojan horses that can hide in systems and graphics processing units (GPUs) for months or years, and malware code that can detect when it has been captured in a sandbox. Meanwhile, firewalls have become less effective as cloud computing and APIs string together data across enterprise boundaries.

Says David Garlan, a professor at Carnegie Mellon University (CMU), “It has become very difficult to pick out malicious events from all the non-ma-

licious events.” This has contributed to an uptick in missed detections, as well as false positives. Likewise, other cybersecurity threats ranging from social engineering techniques like phishing and spear phishing (which target an individual with a realistic-looking message or file) to botnet attacks have become more difficult to pinpoint and block from the network because they use cloaking techniques and alias IP addresses.

Researchers are now taking aim at the problem and attempting to move beyond the traditional cat-and-mouse game that includes whitelists, blacklists, and basic code matching. Enormous advances in computing power, far more sophisticated analytics capabilities, and the emergence of cognitive computing and deep learning, are introducing opportunities to spot malware and attack patterns instantly and adapt systems dynamically.

Cognitive computing scans files and data using techniques such as natural language processing (NLP) to analyze code and data on a continuous basis. As a result, it is better able to build, maintain, and update algorithms that better detect cyberattacks, including Advanced Persistent Threats (APTs) that rely on long, slow, continuous probing at an almost-imperceptible level in order to carry out a cyberattack.

Cybersecurity threats have become more difficult to pinpoint and block from the network because they use cloaking techniques and alias IP addresses.

Deep learning attempts to mimic the human brain and use neural networks to take data and pattern recognition to a level that is practically beyond human grasp. It, too, taps machine learning to update algorithms in an unsupervised or semi-supervised way. The technique is now used for speech and pattern recognition tools that mimic human sight and hearing.

One company at the vanguard of AI is Tel Aviv, Israel-based Deep Instinct, which has introduced security software that uses an artificial neural network (ANN) to digest huge volumes of data and put it to use quickly and effectively.

“The vast majority of new malware consists of very small mutations in comparison to known malware; by some estimates, there’s less than a 2% difference in code,” says Eli David, Deep Instinct’s chief technology officer and a lecturer in machine learning at Bar-Ilan University in Tel Aviv. However, that is enough to throw off most conventional malware detection tools, including those that take aim at zero-day attacks and APTs. For example, the 2014 attack on Sony was a simple and relatively unsophisticated attack, David notes. “It’s just that the malware was new and, as a result, the existing security solutions did not detect it.”

Deep learning strives to reduce, if not eliminate, the endless cycle of manually updating signatures in response to the latest permutation. It examines patterns by training on millions of critical parameters and elements by clusters of graphical processing units (GPUs). These processors operate faster than central processing units (CPUs) and, thus, are critical for training deep neural networks. The firm then distributes the trained neural network to end users, who can use it to identify threats with a lightweight endpoint software agent that installs on servers, laptops, and mobile devices.

Deep Instinct has tested 61 other

Milestones

Computer Scientist Receives MacArthur “Genius” Fellowship

A computer scientist was among the recipients of the 2015 fellowships from the John D. and Catherine T. MacArthur Foundation. The no-strings fellowships are awarded to “individuals who show exceptional creativity in their work and the prospect for still more in the future.”

The winners receive \$625,000 over five years, and people cannot apply for the fellowship; the Foundation simply makes its selections.

Among the 24 recipients of MacArthur Fellowships was Christopher Ré, assistant professor of computer science at Stanford University, who is “democratizing big data

analytics through theoretical advances in statistics and logic and groundbreaking data-processing applications for solving practical problems.”

Ré leveraged his training in databases and knowledge of machine learning to create the DeepDive inference engine, which can analyze data of a kind and at a scale that is beyond the current capabilities of traditional databases. DeepDive analyzes “dark data”—unprocessible data buried in texts, illustrations, images, etc.; it then extracts relationships among entities (such as real-world objects) in the data and infers facts involving those entities. These facts, or assertions, form a

knowledge base, which can then be integrated into an existing database. DeepDive has proved to be more accurate than human annotation, and it can be “trained,” even by users without computer science expertise, to improve the quality of results through simple domain-specific rules and low-level feedback about correct or erroneous predictions.

Ré is also working to overcome the technical challenge of computing the many possible interpretations of each data item with speed and efficiency. He has made significant advances in Incremental (Stochastic) Gradient Descent (IGD) and

has prototyped Hogwild!, which enables data analysis on a multicore machine, and Bismarck, which integrates various analytical tasks into a traditional database system without the need for separate code paths. These improvements have led to the application of DeepDive in a wide range of settings, from scientific laboratories to law enforcement.

After receiving a B.S. from Cornell University and a Ph.D. from the University of Washington at Seattle, Ré spent four years as an assistant professor at the University of Wisconsin at Madison, before joining the faculty of Stanford University.

solutions on datasets of new malware (zero-day) and, according to its internal analysis, Deep Instinct achieved a 98.8% detection rate, compared to 79% for the next-best solution (many of the most famous solutions on the market only identified 20% to 40% of the malware, David says). In addition, the software—which now trains using millions of malware files and data—is more difficult to confuse, once the training process has taken place.

The advantage is clear: “The underlying principle is that you don’t have to understand the reason the software works or why it is detecting malware, it just accomplishes the task.” Consequently, retraining the ANN can take place in near-real time, as conditions and requirements change. David likens it to someone tossing a ball and another person catching it reflexively; “your brain operates in prediction mode and instinct takes over.”

A Healthy Dose of AI

Deep learning is not the only approach that relies on AI and machine-learning methods. Researchers are also exploring a number of other technologies. For example, at Georgia Tech, one initiative revolves around the use of an algorithm that analyzes relationships with peer files using locality-sensitive hashing and graph mining, which clusters risks by probability. The method, now patented and used by Symantec, determines whether a specific file is good or bad. Tests show the approach identifies 99% of benign files and 79% of malicious files a week earlier than other technologies.

Another project relies on touch signature to authenticate a mobile device, but then takes things a step further by constantly comparing the signature with the behavior of the person using the device at any given moment.

At CMU, Garlan and other researchers are pushing the boundaries of AI planning models. “A basic problem with security systems is that you can’t engineer every security feature into a system from the start. You simply cannot anticipate everything that’s going to go wrong,” he says. Consequently, he is studying how to move beyond using AI merely on the detection side and harness it on the repair side.

“Today, repair takes place either at

“We will never move away from the need for human participation. The problem will never completely disappear.”

a very low level or a very high level. The low-level mechanisms tend to be myopic because they are local; they often aren’t able to diagnose what’s really happening because they lack a global perspective. By contrast, human operators are very good at achieving a global perspective on the system, but tend to work slowly and they often aren’t particularly reliable for routine tasks.”

Garlan and other researchers hope to create a middle ground between machine and human that would ultimately lead to autonomous and self-adaptive systems. This means that a security system, when it detects a suspicious event based on code patterns, the time of day, IP address, or numerous other factors, would generate a Captcha or request a two-factor authentication code without involving a human operator.

The ability to view events in a more granular and contextual way could also allow the system to adapt to events and risks in real time. For instance, if a system detects a flurry of attacks or transaction volumes are high, the system might dial up or dial down its level of aggressiveness. This approach could be used for everything from malware detection to overseeing fired and disgruntled employees, and even the inadvertent sharing of sensitive data.

Garlan says the key to moving further into the realm of AI is the ability to process large volumes of data along with continued research on algorithms, machine learning, and neural networks. Many of the answers already exist, he says; it is simply a matter of combining data points, crunching huge volumes of data, and rethinking interfaces to introduce more streamlined and functional machine-human interaction.

“We are going to see systems become smarter and better at analyzing situations through machine learning and AI, including pattern and anomaly detection on the prevention side of the equation, and greater resilience and adaptability and flexibility on the repair side,” Garlan says.

In the end, Daley likens the process of advancing security to castle-building during the Middle Ages: basic moats led to wooden stockades and more advanced stone walls with turrets, which led to flaming weapons. In today’s world, there is a need for data scientists to frame problems and opportunities in new and different ways and incorporate machine-learning methods that can better adapt to the environment while adding new capabilities on the fly.

Regardless of advances in AI and cybersecurity, however, “We will never completely move away from the need for human participation,” says Daley. “The problem will never completely disappear. As long as there are people, there will be cybersecurity risks and threats.”

Further Reading

Tamersoy, A., Roundy, K., and Chau, D.H. **Guilt by Association: Large Scale Malware Detection by Mining File-relation Graphs.** College of Computing, Georgia Institute of Technology. KDD’14, August 24–27, 2014, New York, NY, USA. http://www.cc.gatech.edu/~dchau/papers/14_kdd_aesop.pdf

Saravanan, P., Clarke, S., Chau, D.H., and Zha, H. **LatentGesture: Active User Authentication through Background Touch Analysis.** College of Computing, Georgia Institute of Technology. Chinese CHI ’14, April 26–27, 2014, Toronto, ON, Canada <http://www.cc.gatech.edu/~dchau/papers/LatentGesture.pdf>

Morel, B. **Artificial Intelligence and the Future of Cybersecurity.** In *Proceedings of the 4th ACM workshop on Security and artificial intelligence (AISec ’11)*. 2011. ACM, New York, NY, USA, 93–98. DOI: <http://dx.doi.org/10.1145/2046684.2046699>

Xing Fang; Kocejka, N.; Zhan, J.; Dozier, G.; and Dipankar, D. **An Artificial Immune System for Phishing Detection, Evolutionary Computation (CEC), 2012 IEEE Congress, vol., no., pp.1-7, 10-15 June 2012.** <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&number=6256518&isnumber=6252855>

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

© 2016 ACM 0001-0782/16/05 \$15.00

Coding as Sport

Programming competitions help identify and reward those who excel in coding.

AT MOST COLLEGES, you are likely to find sporting events taking place each weekend throughout the year. In the fall, you have football and lacrosse; in the winter, basketball games and wrestling matches are commonplace, and in the spring, tennis, softball, and track and field events rule the season.

For colleges and universities with computer science programs, however the sporting season runs year-round, and brain matter, rather than muscle matter, is being tested and strained against the competition.

Indeed, sport coding—which is the process of competing in competitions in which participants must write computer code that will solve a problem or problems within a time limit—have become de rigeur for ‘serious’ coders at the secondary school and collegiate levels. These competitions usually feature a problem or series of problems that typically revolve around an algorithm or mathematical structure, and must be solved quickly and efficiently. Like their athletic counterparts, these competitions feature highly driven participants who see these events as a pathway to self-improvement, as well as a bit of fortune.

“A lot of our students come to us and say ‘I have a prestigious degree, I have a good career, but I just love coding,’” explains David Yang, co-founder of the Fullstack Academy, a New York City-based provider of programming courses, and of the Grace Hopper Academy, a tuition-deferred, all-women focused coding program.

“[Participating in coding contests] is a hobby that people can be really passionate about, but also has financial benefits,” Yang adds. “It’s kind of the same reason people do a Tough Mudder [a series of obstacle race/mud run events designed by British Special Forces]. They intentionally punish you, and make you feel that intensity and pain, but out of that you get team bond-



The team from the St. Petersburg National Research University of IT, Mechanics, and Optics came in the first in the 39th Annual ACM International Collegiate Programming Contest (ICPC) World Finals last year in Marrakech, Morocco.

ing, you test your personal limits, and you have fun doing it.”

Getting in the Game

Coding competitions are highly varied, ranging from local collegiate events to large national and international events with thousands of participants. Some of the largest, most popular events include ACM’s International Collegiate Programming Contest (ICPC), Facebook’s Hacker Cup, TopCoder’s Java, C++, and C# competitions, and the Russian Codeforces competitions. As an incentive, many of these events offer cash prizes to the winners; TopCoder, for example, has awarded more than \$70 million in prize money over the past 15 years.

The main drivers of these competitions are easily understandable: they offer the thrill of competing against others, allow participants to benchmark their progress and, perhaps most importantly, provide them a way to showcase their skills to prospective employers.

“Testing your skills and trying to improve your skills is a major part of it,” says Mark Lewis, a computer science professor at Trinity University in San Antonio, TX. Lewis competed in the ACM ICPC when he was a college student, and still occasionally competes in TopCoder events when he has time.

“People say, ‘Hey, if I do really great

here I might win,’” Lewis says. “And maybe, I can go land a good job with a good company as a result of this. There is no doubt that it is part of the attraction.”

Still, some in the coding world have maligned coding competitions, noting these competitions are largely based around completing a task as fast as possible, says Valerie Barr, a computer science professor at Union College in Schenectady, NY, who also is involved with the Women’s International Hackathon run by Microsoft.

“The competition model rewards the kind of thought process that has the ‘aha’ moment, the ‘I can code this really quickly, even if it’s ugly,’” Barr says. “Diversity of thoughts and approaches—the competition model only encourages a small set of those [elements]. Someone who has a thoughtful approach to development, maps out their design, does a lot of desk checking, and does everything with paper and pencil first; that’s not rewarded in a competition model.”

The Gender Gap

Moreover, coding competitions—as well as the computer science field as a whole—are largely dominated by men. “I believe that hackathons are mostly male,” says Julie Gabler, a fourth-year Information Technology and Criminal Justice major at the Rochester Institute

of Technology. “There are, of course, female participants, but teams of four will usually not have more than one female, from what I have seen.”

This male domination appears to be rooted in the way computing was marketed in the early 1980s. According to a segment aired on National Public Radio’s *Planet Money* show, women were very active in the computer programming field from 1970 through 1984. In the early 1980s, the narrative first emerged that computers were for boys, and they were marketed almost exclusively to boys and men.

Furthermore, the notion of the “computer geek” also began to emerge during this period, with TV shows, movies, and video games generally relegating the computing field to boys (think about Anthony Michael Hall’s character in *The Breakfast Club*, versus Molly Ringwald’s character). This led to parents being more likely to buy computers for boys than girls, and many more boys had experience programming prior to entering college. As a result, many young women felt like they were already significantly behind their male peers from the start, and without the proper encouragement and support from peers and faculty, they left computer science programs in droves.

That seems to be changing, thanks to the increasing number of coding events that are designed not to be cutthroat, winner-takes-all affairs. These events are organized around a team-based approach and are focused on solving a real problem, which is why they are often sponsored by large companies looking to crowdsource solutions. Furthermore, events such as these tend to be more inclusive, attracting more women participants, as well as those from outside computer science.

For example, Mount Holyoke College, a liberal arts school for women, hosts HackHolyoke, a 24-hour hackathon founded by two students in 2014 that specifies a 1:1 gender ratio for its participants, and allows and encourages even non-computer science majors to participate. Because it is based on a team-based approach to solving a problem or creating a product, not everyone needs to be an expert, or even a computer science major, in order to participate.

“It’s a very friendly competition to beginners,” according to Jo Martin, di-

rector of employer relations at Mount Holyoke’s career center. “The encouragement is certainly such that anybody that hasn’t participated in the past would certainly be welcomed.”

Indeed, say Eva Snyder and Katie Ho, co-organizers of HackHolyoke, existing hackathons and coding competitions were primarily made up of men, most of whom were hardcore coders. “It was obvious to us that there were very few women there,” Ho says, noting the gender disparity at a hackathon she attended prior to starting HackHolyoke. “We wanted to create this more inclusive, more welcoming environment.”

A Collaborative Approach

Austyn Hill, a Lead Java Instructor at The Software Guild who participated in a coding competition while she was a student at the Massachusetts Institute of Technology (MIT) a decade ago, noted that alternatives to coding competitions such as coding jams and hackathons are often seen as being more attractive events to some coders, since they focus on exploration, cooperation, and collaboration, and less on competition. She competed as part of a team at MIT’s Battlecode, and found that while the prize money and notoriety associated with winning were incentives to participate, the main draw was the opportunity to work with her friends.

“It was a fun project for me and some of my dormmates to work together, and have a project that was independent of grades and everything else like that,” Hill says. “That was probably my biggest motivation.”

Ultimately, whether structured as an intense, single-person coding competition or as a team-based hackathon, these types of competitions are and likely will continue to serve the needs of ambitious students, coding enthusiasts, and the organizations that sponsor them.

Coding competitions are “a place where students can learn to work in teams and develop ideas in a short amount of time,” says Melissa Young, a fourth-year game design and development major at the Rochester Institute of Technology. “Larger companies love the idea of challenges, as it can help them determine skill levels of their applicants and what people’s thought processes are. Quite a few employers actually use online programming tests

as part of their technical interviews.”

Most importantly, regularly competing in any type of programming competition is likely to impress potential employers, as it demonstrates a willingness to address coding challenges outside of a school or work setting.

“I put that I participate in these competitions on my résumé, because I think it shows that I care about computer science beyond completing my coursework,” says Kylie Moden, a computer science major at Trinity University who competed in her third year in the regional ACM programming competition. “If you excel, it’s definitely a good thing for employers to see.”

Morgan Keiser, a second-year computing security major at RIT, agrees, noting one need not be at the top of the leaderboard to reap the benefits of participating in coding challenges and hackathons. “I haven’t been ignored by companies offering mentorship just because I may not be as good as others around me,” Keiser says. “I’ve also made a few connections through hackathons with company recruiters, not for recruiting, but just as professionals.”

Indeed, the best and most lasting benefit of coding competitions and hackathons may not be the results, but the skills that are often most required in the workplace: efficiency, time management, and the ability to collaborate to complete a project.

“It’s just about beginning to breach the fact that coding is not something to fear, because it makes life easier and more fun,” Gabler says. “Like anything, it takes practice, but it is much easier and more rewarding to learn and apply that knowledge with other people and get real-world application to your technical knowledge, no matter what your experience.” C

Further Reading

FromDev coding contest lists: <http://www.fromdev.com/2014/09/best-programming-contest-sites.html>

A gender-balanced college hackathon: www.HackHolyoke.org

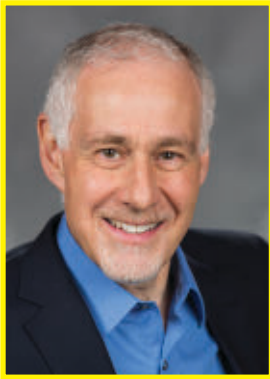
What is a Hackathon?: <http://www.fromdev.com/2014/09/best-programming-contest-sites.html>

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

© 2016 ACM 0001-0782/16/05 \$15.00

ACM

ON A MISSION TO SOLVE TOMORROW.



Dear Colleague,

Computing professionals like you are driving innovations and transforming technology across continents, changing the way we live and work. We applaud your success.

We believe in constantly redefining what computing can and should do, as online social networks actively reshape relationships among community stakeholders. We keep inventing to push computing technology forward in this rapidly evolving environment.

For over 50 years, ACM has helped computing professionals to be their most creative, connect to peers, and see what's next. We are creating a climate in which fresh ideas are generated and put into play.

Enhance your professional career with these exclusive ACM Member benefits:

- Subscription to ACM's flagship publication ***Communications of the ACM***
- Online books, courses, and webinars through the **ACM Learning Center**
- Local Chapters, Special Interest Groups, and conferences all over the world
- Savings on peer-driven specialty magazines and research journals
- The opportunity to subscribe to the **ACM Digital Library**, the world's largest and most respected computing resource

We're more than computational theorists, database engineers, UX mavens, coders and developers. Be a part of the dynamic changes that are transforming our world. Join ACM and dare to be the best computing professional you can be. Help us shape the future of computing.

Sincerely,

A stylized, handwritten signature in black ink.

Alexander Wolf
President
Association for Computing Machinery



Association for
Computing Machinery

Advancing Computing as a Science & Profession

SHAPE THE FUTURE OF COMPUTING.

JOIN ACM TODAY.

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

SELECT ONE MEMBERSHIP OPTION

ACM PROFESSIONAL MEMBERSHIP:

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

ACM STUDENT MEMBERSHIP:

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

- ☐ **Join ACM-W:** ACM-W supports, celebrates, and advocates internationally for the full engagement of women in all aspects of the computing field. Available at no additional cost.

Priority Code: CAPP

Payment Information

Name _____

ACM Member # _____

Mailing Address _____

City/State/Province _____

ZIP/Postal Code/Country _____

Email _____

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

☐ AMEX ☐ VISA/MasterCard ☐ Check/money order

Total Amount Due _____

Credit Card # _____

Exp. Date _____

Signature _____

Purposes of ACM

ACM is dedicated to:

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

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

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

Satisfaction Guaranteed!

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.



Association for
Computing Machinery

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

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

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

Law and Technology

The Internet of Things We Don't Own?

Who will control the 'ordinary pursuits of life' in the digital economy?

CARS, REFRIGERATORS, TELEVISIONS, wristwatches. When we buy these everyday objects, we rarely give much thought to whether or not we own them. We pay for them, we possess them, we wear them or put them in our garages or on our shelves, so we have very little reason to question their legal status or their loyalties. Yet in the last decade or so, we have witnessed a subtle and effective shift to cede control over our purchases, especially when they contain software.

It began with digital content. Movies started telling us where and when they could be played. Soon our music informed us how many devices it would live on. Then our library books began to automatically re-encrypt themselves on the date they became overdue. Now our phones will not allow us to delete certain apps; our televisions listen for when we take a bathroom break, and mattresses can keep tabs on where we slept last night.

The integration of such smart product features with ubiquitous network connectivity, microscopic sensors, large-scale analytics, social informa-

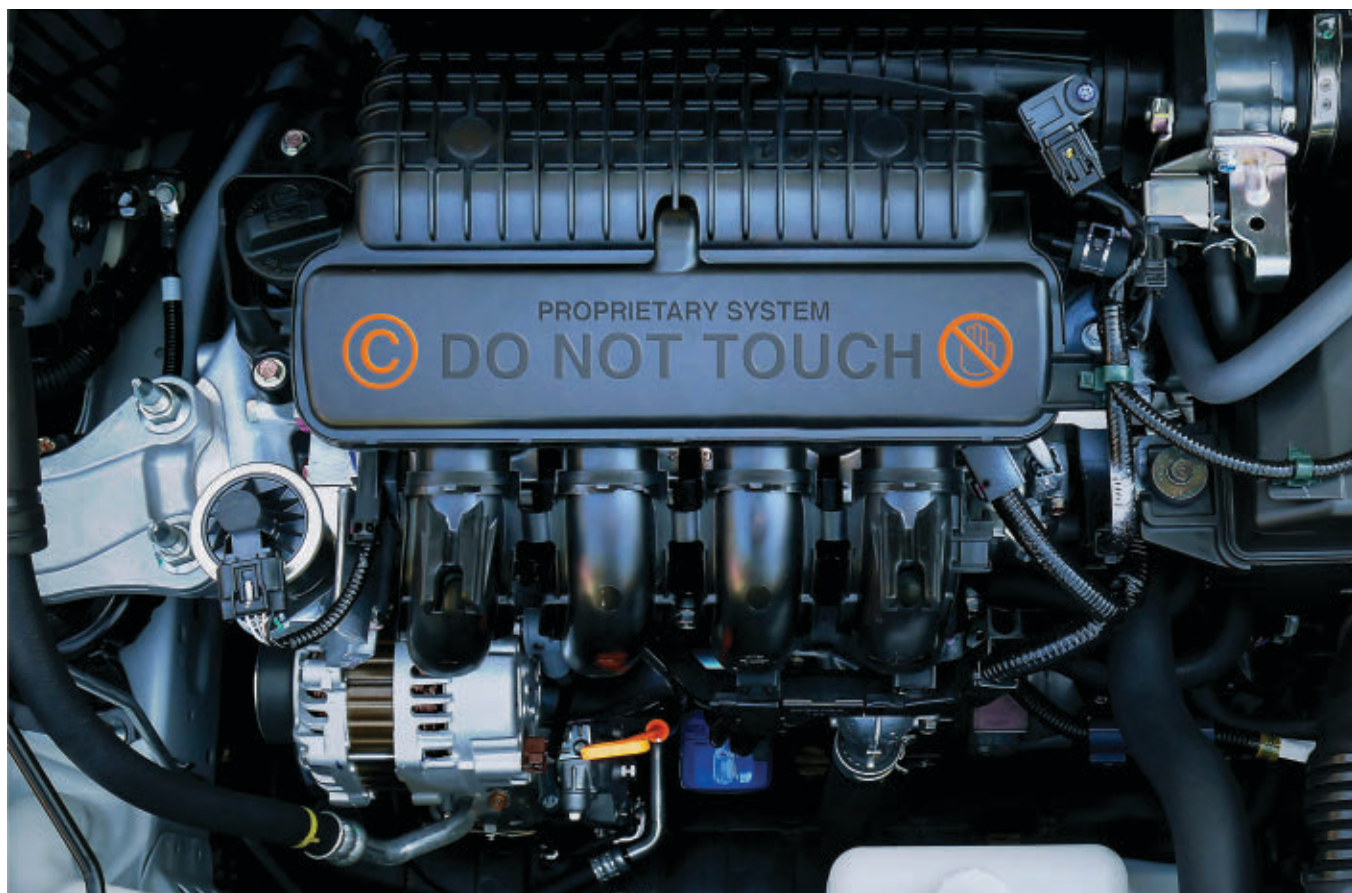
tion sharing platforms, and cloud storage has created a new generation of embedded systems, the Internet of Things (IoT). It is not like the Internet we once knew, and it is not a particularly new idea: embedded computing systems have been around for decades. But the speed of adoption and the diverse capacities of these devices are unprecedented.

The era of IoT has brought more than technological and social shifts. It has also created unusual legal uncertainties. Historically, purchasing

consumer goods, even electronic ones, was largely governed by two areas of law: property and contract. The good was a piece of property. The purchase agreement was a contract. Apart from the occasional equipment rental or lease, if money changed hands, the good went home with its new owner. *Quid pro quo*.

Even goods subject to other laws, such a copyright or patent, generally fell within this framework. As patent or copyright owners sold off individual books, movies, or machines, the law would "exhaust" any remaining intellectual property rights in that particular copy, prohibiting the IP owner, in the words of the U.S. Supreme Court, from interfering with the rights of purchasers to use it "in the ordinary pursuits of life." That meant the purchaser could use the item as she saw fit and then dispose of it, including reselling it, under whatever conditions she chose. These exhaustion rules originated from the long-standing common law regime of personal property, which generally forbids subjecting objects to ongoing restrictions, especially restrictions on resale.

IoT manufacturers and distributors are quietly attempting to shift the rules of ownership.



But that approach is under threat. Digital goods have pushed us away from traditional legal models, and, drawing from the world of software, now come with ubiquitous “Terms of Service” that few if any of us read. Within the dense legalistic language of these documents, IoT manufacturers and distributors are quietly attempting to shift the rules of ownership. For example, many now claim we do not own our phones, our cars, or even our televisions: we are merely “licensing” them. Others assert that when our devices break, it is illegal for anyone other than the manufacturer to diagnose the problem, let alone fix it. And others go even further, claiming any data captured by the device belongs to them and not the users who bought the device and created that data. And while users and consumer advocates have generally pushed back on these assertions, device manufacturers continue to push this view of the world upon us.

The exact origin of this shift is difficult to pinpoint, but one significant moment in its early history was the introduction of the iPhone on January

9, 2007. Steve Jobs told the assembled crowd, “Today, Apple is going to re-invent the phone.” Like nearly every Apple product, the iPhone user experience was carefully choreographed and tightly controlled. What Jobs did not tell the crowd was that Apple’s legal strategy to maintain ownership and control of the devices in our pockets and purses was equally choreographed and controlled.

Eleven days after the iPhone debuted, a group of skillful Apple enthusiasts decided to test its technological and legal limits by “jailbreaking” the phone. This led to a cycle where Apple would upgrade its systems to break the jailbreak and the jailbreakers would upgrade their breaks to free their phones from the upgrade. This battle over who “owns” the device continues to this day, with Apple insisting that “iPhone users are licensees, not owners, of the copies of iPhone operating software.”

As contested ownership over smartphones has become more of a mainstream debate, the battle over IoT ownership has moved into more traditional pursuits of ordinary life.

For example, just last year, farmers found out that many of them may no longer own the equipment they purchased, including even vehicles such as tractors and combine harvesters. Even the iconic John Deere tractor now contains no less than eight control units—hardware and software components that regulate various functions, ranging from running the engine to adjusting the armrest to operating the hitch. When tractors were purely mechanical, farmers could easily maintain, repair, and modify their own equipment as needed. But now, software stands in their way. Tired of losing revenue to industrious farmers who repaired their own tractors or bargain hunters who took their equipment to an independent repair shop, John Deere decided to interpose a software layer between farmers and their tractors, claiming it retained ownership and that farmers merely had “an implied license for the life of the vehicle to operate the vehicle.”

John Deere is not alone. Other vehicle manufacturers including Ferrari, Ford, General Motors, and Mercedes-Benz are finding new ways to use tech-

nology and law to weaken the property interests of drivers. These efforts take a number of forms—DRM that prevents repair and customization, software that monitors and controls your driving, even restrictions on vehicle resale. The car, once a symbol of freedom and independence, is increasingly a tool for control. Modern cars, much like John Deere's tractors, rely on dozens of electronic control units. Access to the software code on those control units is necessary for many common repairs. The code is also crucial if a driver wants to change the default tuning of her vehicle to get more horsepower or better fuel efficiency from the engine, the ambition of a growing group of car purchasers concerned about the environment calling themselves "eco-modders" and "hyper-milers." Yet under the ownership rules of the auto manufacturers, these hobbyists run the risk of becoming copyright infringers.

Such shifts in the battle over IoT ownership are also reaching into the security and safety research communities. As our vehicles incorporate greater computational systems with increased complexity, independent testing of their safety and security will increasingly require access to the copyrighted code inside them. Under the traditional law of personal property ownership, all researchers had to do was purchase a vehicle and then test it; manufacturers had no power to object other than to void the warranty. Despite Ford recalling half a million vehicles due to software glitches, Chrysler recalling 1.4 million vehicles because their infotainment systems were vulnerable to hackers, and notorious scandals such as the Volkswagen's "Defeat Device" that allowed it to cheat on emissions tests for diesel vehicles, we see more and more automakers claiming the code inside our cars is proprietary and access to it without their authorization is illegal. Consumer advocates have pushed back against these efforts, passing a Right to Repair law in Massachusetts and pressuring manufacturers to negotiate a Memorandum of Understanding with aftermarket repair shops and part suppliers that allows those businesses access to diagnostic information for repair and

So what does the law have to say about the question of IT ownership?

replacement purposes. But this does not cover automobile owners.

Nor are our children immune from this shift. Most children have imaginary friends and/or play with dolls. And while we are often surprised at the intensity of these relationships, we have historically understood they were private and ephemeral. Not anymore. Mattel's new WiFi-enabled Hello Barbie doll comes fully equipped with a built-in microphone and a cloud-based machine learning system to "personalize" your child's experience. However, what Barbie won't tell you or your child is that every single word or sound made in her presence will be recorded and transmitted back to Barbie's ML master archive for research purposes. In order to discover that, you would have to read her online Privacy Policy and Terms of Use. With the introduction of this capacity in our children's toys and other home devices such as the Nest thermostat and the Samsung "listening" Smart-TV, the sense of privacy and autonomy we used to enjoy in our homes and with the objects we owned has become yet another contested space in the IoT era.

So what does the law have to say about the question of IoT ownership? To date, neither the courts nor Congress have resolved the question. In general, the courts are split on the exact rules for who "owns" embedded copyrighted media, including software. Some have taken a somewhat technocratic approach, simply deferring to whatever words the maker puts in her license or TOS, regardless of whether or not those words accurately reflect the realities of the transaction. Other judges, however, have been more cautious, recognizing that

consumer expectations play an important role in transactions, especially those involving physical objects. The Supreme Court has come close to weighing in on the issue in some of its recent patent and copyright cases, but has not given us a definitive rule.

Even the Copyright Office has avoided opining, for example, choosing to grant smartphone jailbreakers an exemption from anti-circumvention liability under copyright law's fair use doctrine instead of declaring them owners with the right to modify embedded software. Recently, Congress has taken more action with Rep. Blake Farenthold introducing the You Own Devices Act (YODA), both houses beginning to examine the possibility of updating the copyright exhaustion rules for the digital age, and Senators Grassley and Leahy specifically asking the Copyright Office to analyze "how copyright shapes our interactions with software in things we own." The Commerce Department also recently issued a White Paper expressing concerns for consumers and the market if IoT manufacturers begin placing restrictions on the freedom to resell devices. But while many of these voices are asking good questions, none have provided the answers we need.

To find the answers, we will need to have a more open and honest conversation about ownership—in the courts, in Congress, and in the technical communities that are designing the IoT ecosystem. Hiding these conflicts and questions in shadowy TOS and embedded firmware code will only further confuse consumers and courts and ultimately complicate instead of clarify the rules we want when it comes to our ability to enjoy the ordinary use of these objects, including our ability to use them privately, to customize them to our needs, and even to part with them as we please. **C**

Jason Schultz (SchultzJ@exchange.law.nyu.edu) is Professor of Clinical Law at New York University's School of Law.

The argument in this column is further explored in the author's forthcoming book *The End of Ownership: Personal Property in the Digital Economy* (MIT Press, Fall 2016), with co-author Aaron Perzanowski.

Copyright held by author.

Education

Beyond Blocks:
Syntax and Semantics

How the future of general-purpose programming tools could include blocks-based structured editing, and how we should study students transitioning to text-based programming tools.

MILLIONS OF STUDENTS learning to program today begin with blocks-based systems. Blocks are visual representations of program code that add shape, color, and drag-and-drop editing to concrete syntax (see Figure 1). Appearing in structured editors as early as 1992,⁶ blocks offer a number of affordances: prevention of syntax errors (see Figure 2), type hinting (the diamond-shaped slot in Figure 1 only fits a Boolean-valued expression), and API discovery (block editors typically provide a palette of available objects and methods). Novices benefit from these affordances to overcome some common challenges of learning to program, and there is modest evidence that blocks can improve student learning compared to text.^{5,7}

Most users of these beginner-specific systems will probably never want to progress beyond them, but what about those who do? Will they stick with blocks or migrate to text? Will blocks-based editing become a standard for industrial and academic language developers to consider, or will learners' progress from introductory environments necessarily involve leaving blocks behind?

Mainstreaming Blocks

Several research projects are exploring what general-purpose programming tools with blocks-like features could



Three middle schoolers program a BlocklyTalky synthesizer to play part of a pop song.

be like. The Snap! Project investigates how blocks can scale up from simple to complex programs, such as those involving complex data structures, user-created abstractions, object-oriented programming, interaction with the Web, or even high-performance computing techniques.⁴

Block and text hybrid environments can improve the efficiency and usability of both as shown in multiple current efforts. For example, Greenfoot 3 enables users to manage long segments of Java code through block-like elements called frames, while preserving the ability to readily edit the frames' code within using the key-

board and text, thereby eliminating one of the common complaints about blocks, which is they are cumbersome for advanced programmers. Work like this may soon illustrate how the future of general-purpose programming tools could include blocks-based structured editing.

Meet the New Code, Same as the Old Code

Editing interfaces—and programming tools and culture generally—have proven surprisingly resistant to change. Despite being 40 years old, vi is still a very popular text editor among programmers, even while more user-friendly

alternatives abound. Many programmers, including novices, are resistant to graphical interfaces that enhance the experience of programming: one study of entry-level high school programming students found that while the students overwhelmingly perceived blocks-based programming as easier, they also saw it as less powerful, slower, and inauthentic.⁸ African-American students' preferences for blocks or text vary sharply depending upon their long-term career goals, with those who identify programming as their career goal preferring text over blocks.³

At least for now, it is impossible to participate in existing socio-technical ecosystems of general-purpose programming without becoming fluent in reading and editing text, including text written by others. Regardless of whether blocks become the next general-purpose standard, learners in the

near future will need to transition from blocks toward text.

Blocks → Text; Novice-Specific → General-Purpose

Block-to-text translation tools can improve students' ability to learn traditional languages after using blocks.² But no published empirical research has actually examined the experiences that students have as they transition from blocks to text, including the challenges they face as they progress beyond the walled gardens of novice-specific languages and frameworks.

Current efforts to support transitions from blocks to text are focused on systems building, rather than learning research, and involve transformation of blocks into isomorphic text equivalents. A recent workshop at IEEE Symposium on Visual Languages and Human-Centric Computing, *Blocks and Beyond*, included demonstrations of several tools that involved 1:1 translation of block programs into text equivalents. But this focus on blocks versus text (and blocks *to* text) is a misallocation of our attention. Even though learning to use the concrete syntax of general-purpose programming languages is important to nascent developers, it is not the most difficult problem they face.

With few exceptions, block-based programming environments couple the *syntactic* support of blocks with the *semantic* assistance of purpose-built be-

ginner-friendly APIs. For example, App Inventor drastically simplifies Android development through abstractions that permit creating mobile applications in just a few lines of code. But if learners wish to go beyond what is possible within App Inventor's wrappers of the Android SDK, they must simultaneously abandon blocks for text and learn Java and native Android. A text version of App Inventor's API would only address a limited slice of the problems learners face when they wish to develop more powerful programming capabilities. Focusing on transitions between isomorphic block and text languages (for example, Pencilcode, shown in Figure 2) ignores the semantic problem.

We need research that investigates how we can support learners' transitions from syntactically and semantically assisted novice-specific programming tools into the less-assisted world of general-purpose languages and platforms.

Supporting Conceptual Deepening

We are investigating how to design programming tools that not only support transitions from blocks to text, but that also provide gradual exposure to the semantics of a general-purpose programming environment. To do so, we have created BlocklyTalky, an open source educational programming environment for networked physical devices. It runs on low-cost computing devices like the Raspberry Pi and enables users

Figure 1. IF/ELSE block from Scratch.

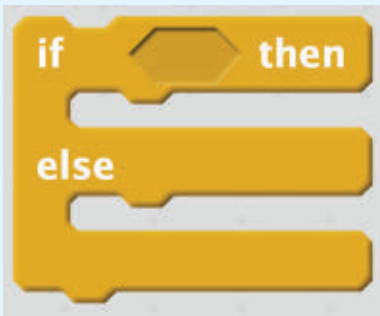


Figure 2. Pencilcode's block-based HTML editor only allows syntactically correct code and permits toggling between isomorphic text and block representations. Image from Aggarwal et al.¹

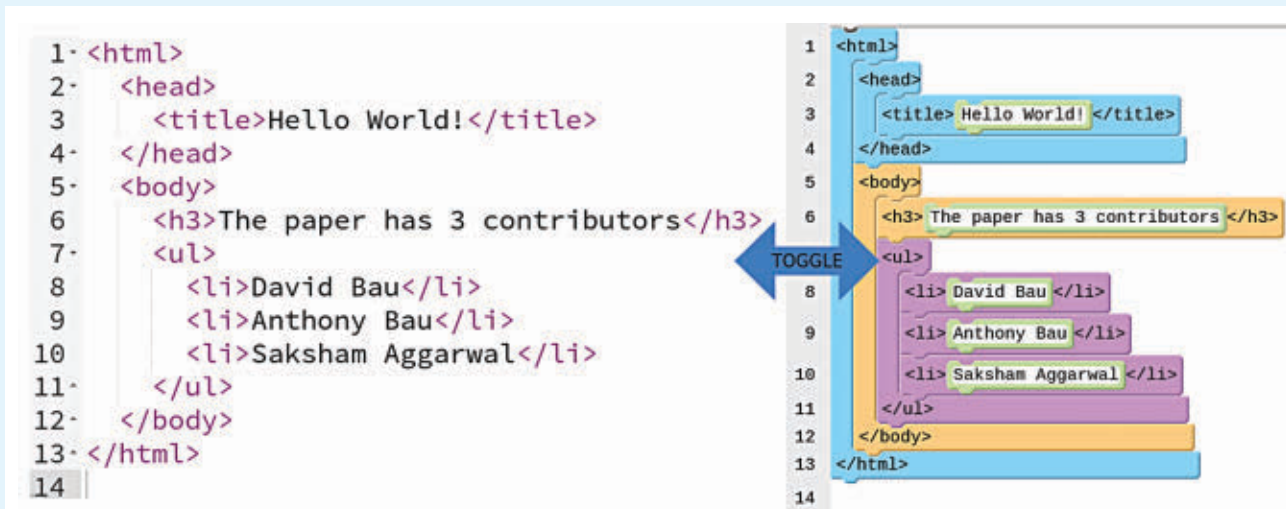


Figure 3. BlocklyTalky event handler.



to create a wide variety of inventions, from computer music systems to Internet of Things projects. Users can begin with a semantically simplified block environment and then transition to a more powerful and flexible text experience. Block-to-text translation eases this task by enabling users to begin their work in text with semantically equivalent and syntactically similar versions of their block programs, but it is only through programming activities that go beyond those isomorphisms that learners access the richness of the general-purpose environment.

Consider an example: Authoring block-based programs in BlocklyTalky involves using simple abstractions for problems like handling physical inputs and communicating over the network with other devices. Figure 3 shows an event handler that asynchronously sends a message (*tickle*) to another device (*elmo*).

BlocklyTalky users can access textual versions of their block programs in a Domain Specific Language (DSL) that we have created to encapsulate event handling, scheduling, networking, and other facets of physical computing that are semantically complex. For example, the blocks in Figure 3 are rendered in the DSL as shown in Figure 4.

We are just beginning research to understand how this block-to-text translation can support the beginnings of student-motivated pathways into additional semantic and syntactic complexity. Our approach is grounded in other researchers' findings that students perceive text-based programming tools as ways to access more powerful capabilities. We respond to these perceptions by limiting what users can do in blocks and their isomorphic textual equivalents, and then offering structured opportunities to gradually learn a more syntactically and semantically rich space.

We envision that students will use our environment in three stages. First, they will work mostly in blocks, and then begin comparing their own block programs to textual equivalents. Because the semantics of the blocks and text will be identical, the comparisons will help them to understand the syntax differences. Next, we will encourage them to tinker with the text programs—that is when they will face the changed semantics of the text language. Finally, they will do things in text (like calling functions in text-only libraries) that they cannot do in the blocks we provide.

Students' progress through these stages will primarily be motivated by their own creative goals. For instance, a student building a device to provide an ambient representation of the outside temperature might want to get data from a Web service that provides weather information. To do so, they would write textual code to call an HTTP library and then parse the data. Though we could add HTTP blocks to make this easier, we have chosen not to. Students already believe a major reason to use blocks over text is to create more powerful programs. We leverage this belief in our approach by limiting the functionality available in blocks and encouraging students to try text in order to accomplish their goals. In this way, learning to write text feels like a natural method of accomplishing of students' own goals, rather than something we impose on them. We will study students' thinking as they work through the three stages just described, identifying areas of confusion along the way and developing new technological and pedagogical supports in response.

Conclusion

Blocks have quickly proliferated within programming tools for novices, of-

Figure 4. Textual version of Figure 4 in the BlocklyTalky DSL.

```
when_sensor "PORT_1" == 1 do
  send_message("tickle", "elmo")
end
```

fering many kids greater ease of entry to computing. While they are helpful, their use creates a new challenge for learning: learning to *not* use blocks. But rather than studying the blocks-to-text transition in isolation, we need research approaches that take a more holistic view, investigating how the blocks-to-text transition is a part of syntactically and semantically rich development. We encourage others doing research in this area to investigate how their work on blocks and learning can take a more comprehensive view, looking beyond blocks and into the richer semantic challenges of learning to program. 

References

1. Aggarwal, S., Bau, D.A., and Bau, D. A blocks-based editor for HTML code. Position paper for the Blocks & Beyond workshop at IEEE Symposium on Visual Languages and Human-Centric Computing (2015).
2. Dann, W., Cosgrove, D., Slater, D., Culyba, D., and Cooper, S. Mediated transfer: Alice 3 to Java. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (2012), ACM, 141–146.
3. DiSalvo, B. Graphical qualities of educational technology: Using drag-and-drop and text-based programs for introductory computer science. *IEEE Computer Graphics and Applications* 6 (2014), 12–15.
4. Feng, A., Tilevich, E., Feng, W. Block-based programming abstractions for explicit parallel computing. A VL/HCC 2015 Workshop. In *Proceedings of the Blocks and Beyond: Lessons and Directions for First Programming Environments*. (Atlanta, GA, Oct. 2015).
5. Lewis, C.M. How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE '10)*. ACM, New York, NY, 346–350; DOI=<http://dx.doi.org/10.1145/1734263.1734383>
6. Minör, S. Interacting with structure-oriented editors. *International Journal of Man-Machine Studies* 37, 4 (Apr. 1992), 399–418.
7. Weintrop, D. Minding the gap between blocks-based and text-based programming. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (2015), ACM, 720.
8. Weintrop, D. and Wilensky, U. To block or not to block, that is the question: Students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (June 2015), ACM, 199–208.

R. Benjamin Shapiro (ben.shapiro@colorado.edu) is an assistant professor in the ATLAS Institute, the Department of Computer Science, and (by courtesy) the School of Education at the University of Colorado, Boulder.

Matthew Ahrens (Matthew.Ahrens@tufts.edu) is an M.S./Ph.D. student in computer science at Tufts University, Medford, MA.

The authors' work is supported by NSF Awards 1418463 and 1453201, the NCWIT Academic Alliance Seed Fund, and LEGO Education.

Copyright held by authors.

Economic and Business Dimensions

Do Patent Commons and Standards-Setting Organizations Help Navigate Patent Thickets?

Examining the institutions underlying the patent system for information and communication technologies.

THE IMPLICATIONS OF the patent system for information and communication technologies (ICT) has been a focus of debate in recent years. One representative example is the smartphone dispute between Apple and Samsung, which led to over 40 international lawsuits over their design and utility patents between 2012 and 2014.

Patents are intended to provide incentives for innovation by granting inventors temporary monopoly rights to profit from their inventions. Therefore, the patent system has the potential to promote new technology development and economic growth. However, ICT products often consist of interlocking inventions. If most of them are protected by intellectual property rights (IPRs), patent thickets could exist: dense, overlapping webs of patents that any company trying to commercialize new technology must navigate. An analysis by Thomson Reuters concluded Apple held 1,298 patents related to its iPhone. This included 416 patents covering general smartphone technologies, 232 patents for user interface, and 149 patents for screen and image display.²

The presence of patent thickets may create challenges for ICT producers.



When introducing a new product, a firm must identify patents its product may infringe upon. When these patents are dispersed among many owners, identifying relevant patents can be difficult and licensing negotiations with patent holders can be costly and time-consuming. Alternatively, if the firm does not obtain necessary licenses, it

may face litigation that could lead to damages and potential injunction. Firms with an existing patent portfolio can navigate the patent thicket by cross-licensing their own patents with other patent holders. However, this may create incentives to engage in “defensive” or “strategic” patenting to defend against future lawsuits, which it-

self may increase the size of the patent thicket. Further, this cross-licensing strategy may not be feasible for small start-up firms who do not possess significant patent portfolios, and who cannot easily acquire them.

Some firms have attempted to proactively address the costs of identifying IPRs and obtaining licenses by opening their IPR portfolios to outside firms. IBM donated 500 patents to the “Patent Commons Project” (commonly referred to as “The Commons”) in 2005, stating it would not enforce the patents against firms developing open source software (OSS). IBM, Sony, Bosch, and DuPont created the “Eco-Patent Commons” in 2008 to promote environmentally beneficial innovation: 100 patents have been pledged to this commons by 13 firms. Tesla Motors opened all of its electric car patents to other firms in 2014 and pledged not to launch patent lawsuits against anyone who wishes to use the technology protected by these patents “in good faith.”

Firms might be willing to allow others to use patented technologies royalty-free to stimulate product growth in complementary markets. IBM’s donation to the “The Commons” might spur innovation in OSS that promotes IBM server and service sales. Through opening up its patents to competitors, Tesla might be able to build a common platform to boost the demand for electric vehicles. If that is the case, one would expect the donated patents are valuable and could encourage innovation by outside firms. However, some have argued these types of donations are merely publicity campaigns. If this is the case, then their economic impact will be more limited.

We examined 2,054 start-up software firms to ascertain the economic impact of The Commons on new OSS products between 1999 and 2009.³ In January 2005, IBM pledged access to its 500 patents to any individual, community, or company working on or using OSS. Subsequent to IBM’s action, several other firms pledged an additional 29 patents to The Commons. These patents cover technologies in a range of markets such as operating systems, databases, and software development tools. Standard metrics show the quality of the patents in The Commons was similar to those of other patents

held by IBM and other firms in similar technology areas, suggesting these patents had real value to IBM. Patent contributions to The Commons stimulated new OSS product introductions by these start-ups for two possible reasons: by using patents in The Commons, start-ups faced lower invention and licensing costs; more importantly, the contribution behavior might signal patent contributors’ commitment not to enforce IPRs against OSS firms, reducing litigation threats.

We found contributions to The Commons had a particularly large impact on stimulating new OSS product introductions in technology environments with denser patent thickets. In such environments, innovation builds heavily upon prior discoveries, and the boundaries of potential blocking patents are usually blurry. It is more difficult for start-ups to invent around existing patents and to determine what patents its needs to license. The promise of the patent commitments to The Commons seemed to be particularly valuable for start-up firms operating in these environments.

Standards-setting organizations might aid with patent thickets as well. In ICT industries, products are usually based on compatibility standards. A study suggests a laptop computer embodies 251 standards.¹ Besides reducing technical uncertainty, the formal standardization process within standards-setting organizations often reveals the holders of essential patents to standardized technologies. Many standards-setting organizations also require these patent holders to license patents on reasonable and non-discriminatory terms. Therefore, formal standardization might decrease IPR uncertainty by identifying relevant patent holders and clarifying licensing terms.

Formal standardization can also increase legal risks. Incorporating technology into a compatibility standard makes it more difficult to use alternative technologies, so standards-essential patent holders can request large licensing fees. For example, Rambus did not disclose its patents in the JEDEC standard for memory chips until the standard was widely adopted. This behavior can increase the costs of using the standardized technologies.

We studied formal standards developed by the Internet Engineering Task Force (IETF) between 1994 and 2004, a period of rapid growth in the commercialization of the Internet.⁴ Standards-using firms were less likely to patent defensively in highly standardized areas. This suggests formal standards might reduce IPR uncertainty within this setting. However, IPR disclosure and licensing rules in use at standards-setting organizations vary widely, and changes in these rules can influence the outcome of standardization.

Although patents create incentives for firms to conduct innovation, patent thickets can create significant costs for ICT producers, particularly for start-up firms. Patent commons can reduce some of the costs of patent thickets by making certain technologies available to use royalty-free, and by signaling that patent holders will not enforce IPRs against certain types of uses. Standards-setting organizations may help to navigate patent thickets in a different way, by clarifying patent holders and reducing uncertainty surrounding licensing terms. The effects of both of these institutions will depend on their rules. For example, technology users will be able to navigate patent thickets more easily when standards-setting organizations set clear IPR disclosure and licensing rules that reduce frictions between standards-essential patent holders and users. ICT producers will need to understand these rules thoroughly to understand the implications of these institutions. 

References

1. Biddle, B., White, A., and Woods, S. How Many Standards in a Laptop? (And Other Empirical Questions), 2010; http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1619440.
2. Thomson Reuters. Inside the iPhone Patent Portfolio: Data Offers Glimpse of Battery-Free Future, Insight into Smartphone ‘Patent Wars’, 2012; <http://ip-science.thomsonreuters.com/m/pdfs/iphone-report.pdf>.
3. Wen, W., Ceccagnoli, M., and Forman, C. Patent commons, thickets, and open source software entry by start-up firms. *Management Science*. Forthcoming.
4. Wen, W., Forman, C., and Jarvenpaa, S. Standards, intellectual property rights, and strategic patenting: Evidence from the IETF, 2015; http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2709645.

Wen Wen (wen.wen@mcombs.utexas.edu) is an assistant professor in the Department of Information, Risk, and Operations Management at the University of Texas McCombs School of Business.

Chris Forman (Chris.Forman@scheller.gatech.edu) is the Alton M. Costley Chair of Sales and Management at the Scheller College of Business at the Georgia Institute of Technology.

Copyright held by authors.

Historical Reflections

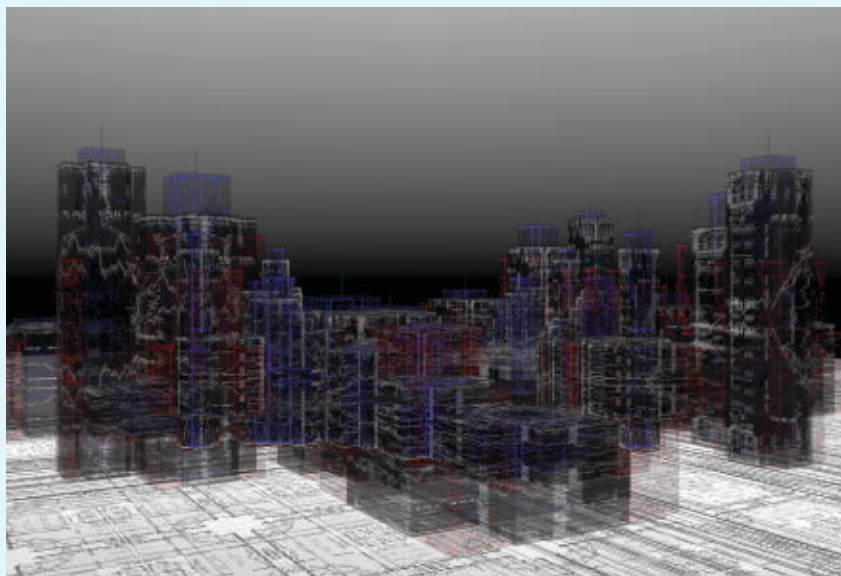
Preserving Hybrid Objects

Brutal lessons from contemporary art.

IN AUGUST 2015, a minor furor broke out on the SIGCIS and Humanist discussion forum about the merits or otherwise of Tara McPherson's essay "Why Are the Digital Humanities So White? or Thinking the Histories of Race and Computation."

McPherson's attempt to knit together "discussions of race (or other modes of difference) with our technological productions within the digital humanities (or in our studies of code)," depends on drawing parallels between the development of MULTICS (and then UNIX) and the more or less contemporaneous Civil Rights events of the 1960s. Her case rests strongly on the notions of modularity and encapsulation, which she presents as something akin to code apartheid. McPherson opines "I am not arguing that the programmers creating UNIX at Bell Labs and in Berkeley were consciously encoding new modes of racism and racial understanding into digital systems." She states her purpose as showing "the ways in which the organization of information and capital in the 1960s powerfully responds—across many registers—to the struggles for racial justice and democracy that so categorized the United States at the time." In pursuit of this goal McPherson sketches two historical fragments drawn from the history of the 1960s: the first is a potted history of the development of UNIX, "well known to code junkies and computer geeks," while the second, famil-

Figure 1. Michael Takeo Magruder's "Data Plex" series of artworks utilize live data feeds from real-life scenarios to generate three-dimensional geometry and textures in real time; <http://www.takeo.org/nspace/ns031/>.



iar to "scholars of culture, of gender, and of race like the members of the ASA"^a concentrates on the "struggles over racial justice, [and] antiwar activism" going on at the same time.

McPherson's suggestion is that rather than being viewed as parallel but independent, these fragments should be seen as "deeply interdependent." This would be a much more interesting suggestion if only it were accompanied

by an evidence-based argument showing grounds for believing the developers of UNIX were in any way following a racist agenda, or were demonstrably responding to racist notions. McPherson's opinion seems to rest entirely on the notion that because UNIX was developed in a time and place where racism was foregrounded, features of the UNIX operating system must be reflective of that wider social context. While not completely uncontentious, it is unremarkable to assert that all things may be thought of as interdependent. The interest lies in showing the degree of 'influence' is more than marginal, and

^a McPherson seems unaware of, or unconcerned about, the inescapable value judgments that appear to underlie this way of describing matters.

this requires evidence of a stronger kind than is offered in McPherson's essay.

It is hardly surprising, given the intentionally provocative and polemical character of the original essay, together with the privileged status she gives to explanations of features of MULTICS/UNIX that depend on attributing covert or unconscious racism on the part of their designers, that some contributors to the discussion forum conversations expressed themselves in somewhat intemperate tones.

I do not find McPherson's 'post hoc ergo propter hoc' position persuasive. Her essay is underdeveloped, and fails to consider many other explanations for the single technical feature of UNIX onto which she latches. As it stands, she offers no more reason to believe UNIX is racist, than it is sexist, and while either or both of these claims may have merit, she does nowhere near enough to convince, but more than enough to provoke. She seems content to note 'parallels,' some of which appear contrived, without demonstrating linkage or causal connection. One of the issues is certainly that different disciplines see the world in different ways, and this extends to include notions of what counts as an argument, or a 'proof'.

The heated nature of the discussion set me thinking about the extent to which work in the 'Digital Humanities' which should be marked by interdisciplinarity and collegiality, is so often characterized by groups separated by a common cause. My own experience of working with people from different disciplines, as well as my personal interdisciplinary journey, has been almost entirely productive, and I have learned much from colleagues whose perspective and intellectual direction of travel stands in marked contrast to my own.

By way of example, some years ago I was lucky enough to spend time exploring preservation issues with the contemporary artist Michael Takeo Magruder, and the examples that follow are drawn from his corpus of work.

A great deal of progress has been made over the last decade in the field of digital preservation, and to all extents and purposes we may regard the problem of simple bit storage as solved. However, the world does not stand still, and new problems arise while the old

Figure 2. Michael Takeo "Magruder's Data Flower (Prototype I)" creates an endless cycle of ephemeral synthetic flowers; <http://www.takeo.org/nspace/ns034/>.



problems are being addressed. Simple digital objects such as ascii files represent an ever-decreasing proportion of the overall problem space, and attention is still required in the domain of "complex digital object" preservation.

As it happens, many of the problems digital preservation is currently facing are well illustrated by looking at the difficulties involved in preserving contemporary artworks having a digital component. Artists have always been early adopters of new technologies, and have taken inspiration from the opportunities digital computing has provided for novel forms of artistic expression. The result is that contemporary art now represents one of the 'bleeding edges' of digital preservation.

It is useful to begin by noting something that is obvious to artists, but perhaps not so much to technologists: even when fully digital, artworks are not 'only' digital. They are not to be understood merely in terms of hardware

and software, and any preservation approach that does not acknowledge this is doomed to failure. In any attempt to preserve and make accessible the present for the future, it is vital to comprehend the full context of that which we are attempting to bequeath to future generations. This is often expressed in preservation circles in terms of preserving 'significant' properties for particular 'stakeholder' communities.

Of course, many contemporary artworks are not fully digital in the first place, but are combined with analogue elements. For example, Jose Carlos Martinat Mendoza's sculpture, "Brutalismo," is a scale model of the Peruvian military headquarters (itself an example of 'brutalist' architecture). The artwork incorporates a computer that searches the Web for references to 'Brutalismo/Brutalism'. The search hits, which capture a variety of examples of political and architectural brutalism, are printed on small pieces of paper that are spat out onto the gallery floor. "Brutalismo" is a 'hybrid' object, drawing its expressive power, and its artistic value, from a combination of digital and analogue elements, neither of which can fully be understood in isolation. Not only do we need to be sensitive to the object itself when devising a preservation strategy, but attention must also be paid to how it was situated in the gallery. Placing the artwork against cheery primary colors conveys something different than does a muted grey environment. A preserva-

Contemporary art now represents one of the 'bleeding edges' of digital preservation.

tion approach that preserves only the digital elements of “Brutalismo” (even if we extend this to include the computer hardware) simply misses the point. Similarly, the artwork is more than its physical embodiment, however arresting that may be. An additional set of difficulties arises from the incorporation of live data.

Another striking example of the use of live data is Magruder’s “Data Plex” series of artworks that use live data feeds from real-life scenarios to generate three-dimensional geometry and textures in real time, creating virtual realms that refract ever-changing, volatile forces in and upon the real world (see Figure 1). The technology that drives “Data Plex” is a combination of server-side Java and client-side Virtual Reality Modeling Language (VRML) and, in preservation terms, this makes it relatively easy to capture ‘screen grabs’ and to save the .wrl geometry files that might be thought of as constituting the complete environment.

However, it is far from clear that a series of snapshots are capable of capturing fully the artistic characteristics of the artwork. “Data Plex (economy)” was conceived in the immediate aftermath of the 2008 global financial crash, and was intended by the artist to allow audiences to interact with a live ‘embodiment’ of the financial market, and to witness its volatile fluctuations in real time. It might be possible to capture this unique moment in time by storing a record of the fluctuations in the Dow Jones Industrial Average that ‘drove’ the artwork in 2009, and preserving this information along with the rest. Regrettably, this approach is not entirely unproblematic as the original installation interacted with the live Web, not with a fixed dataset, and the artwork would require reprogramming to do otherwise. In taking this approach to preservation, we would be privileging one particular period in the ‘life’ of the artwork rather than capturing a history—some of which is still to be written.

Another of Magruder’s artworks, “Data Flower (Prototype I),” creates an endless cycle of ephemeral synthetic flowers using a combination of VRML to define the basic flower geometry, attenuated by (pseudo) randomized parameters that produce subtle muta-

It is not always easy to be sure what constitutes the object of preservation.

tions within the petal formations, and ensure each flower develops differently (see Figure 2). Finally, surface textures are produced by sampling the 100 most recently uploaded Flickr photographs that have the tag ‘flower’. These are stored in a temporary database, from which an image is (pseudo) randomly selected and applied across the developing floral geometry. The final on-screen appearance is therefore the result of a combination of algorithmic calculation, (pseudo) randomness, and an entirely unpredictable and ever-changing set of external images. As with “Brutalismo,” attempting to reconstruct even a fraction of the “Data Flower’s” timeline is made difficult because not all of the information is stored internally; the internal database being overwritten every day, and may not be available from external data sources at the time when preservation is attempted.

The situation becomes further complicated when some element of interactivity is incorporated, for example, when the behavior of an artwork is affected by the number of people who are within a gallery at a given moment, or the exact time of day when a viewer passes the artwork. In cases such as these, it is simply not feasible to preserve faithfully the actual behavior of an artwork for future examination.

These, and similar considerations, should make us consider very carefully what exactly it is we are trying to preserve, and for which stakeholder community? It is not always easy to be sure what constitutes the object of preservation. While preserving a series of ‘snapshots’ may be perfectly acceptable to some stakeholder groups, it is unlikely to satisfy others.

Interactivity always presents the preservationist or conservator with difficulties and may be considered as a ‘problem case’ in its own right or as

an additional complication when combined with other problems such as object hybridism.

The widespread and increasing use of social media adds yet another dimension. Platforms like Second Life offer interesting creative possibilities for artists to explore. In addition to capturing something of the current Zeitgeist, social media platforms permit artists to collaborate with each other and with users in ways that are otherwise not open to them. Of course, collaborative working can create problems in understanding clearly who “owns” an artwork, and how to acknowledge the contributions of everyone who played a part in the creative process. In this context, it is well worth reading Jerry McDonough et al.’s very instructive report on the Preserving Virtual Worlds project (<http://hdl.handle.net/2142/17097>), which draws attention, in Chapter 7, to the difficulties that arise in trying to preserve an ‘Island’ in Second Life. Even though the team had at their disposal tools that should have enabled them to achieve more or less complete preservation, they were, primarily as the result of ‘rights’ issues, only able to manage “extremely partial and static representations of the original.”

Working with contemporary artists has made clear to me that any inclination one may have to believe that preserving artworks is primarily a matter of developing an appropriate set of software tools and workflows, is quite mistaken. From the artistic perspective, the hardware and software aspects of artworks are clearly important, indeed without them then works would not exist. However, the essence of the object of preservation lies somewhere beyond these components, and calls into question any technologically deterministic approach to preservation. This is a lesson that is well worth extending into other areas of preservation.

I have not hinted at how the challenges I have highlighted might be partially or fully addressed, but will return to this topic in a future column. 

David P. Anderson (cdpa@btinternet.com) is Professor of Digital Humanities at the Centre for Research & Development (Arts)/Cultural Informatics Research Group, University of Brighton, U.K.

Copyright held by author.

Viewpoint

How to Increase the Security of Smart Buildings?

Surveying unresolved security problems for automated buildings.

IN RECENT YEARS, known vulnerabilities such as in the Tridium Niagara AX Framework,^a Siemens' energy automation device,^b or Vaillant heating systems^c put a spotlight on the security of smart buildings. What is needed to increase the security of smart buildings and what research aspects did we forget to cover in recent years?

Smart buildings are automated buildings designed to save costs, to increase our safety and comfort, and to be environmentally friendly⁴ while being able to interact with other smart things and the energy grid. When we think of a smart building, we may first think of a home in the suburbs or of a high-tech flat for techies. This impression is, however, only a small pixel in the big picture of smart buildings. While smart homes become an increasingly integral part of today's cities and villages, smart buildings can be found in other areas, too. In factory buildings, building automation technologies handle the physical access control to prevent undesired visitors; in greenhouses, building automation equipment ensures the temperature, humidity, and illumination are kept at the optimal level for the growth of plants; in server rooms, we find



equipment that handles the air conditioning to prevent servers from overheating; and in airports, we find automated building components almost everywhere, ranging from elevators to smoke detectors and fire alarms to electric window shutters.

When mentioning such elementary components of smart buildings, we can already deduce that building automation is more than current hype. Instead, such features were integrated into buildings for decades.

Indeed, building automation existed for a long time before the first electric components were integrated into the buildings of the 1960s.⁶ Only in recent years, vendors, due to market demand, started to connect their automated buildings to the Internet to provide new services. These services, such as remotely controlled heating or outsourced video surveillance, do not simply become popular because they are “nice to have.” Instead, society will depend on these services to a

a <http://www.wired.com/2013/02/tridium-niagara-zero-day/>

b <http://www.infosecurity-magazine.com/news/siemens-energy-authentication/>

c <http://www.hotforsecurity.com/blog/vulnerability-in-vaillant-heating-systems-allows-unauthorized-access-5926.html>

larger extent every day. This fact is especially applicable to aging societies such as in Japan or Germany. Ambient assisted living provides technology to care for the elderly, for example, via sensor-equipped floors that detect when people fall down and report this event to on-call personnel over the Internet.

In this Viewpoint, I highlight six unsolved problems for smart building security. While several other non-technological problems (such as cyber insurance for smart buildings) exist, the selected issues appeared on a reoccurring basis in recent discussions with partners from academia and industry; both stakeholders desire solutions for each of these problems.

Internet-based Communications.

First, these now Internet-connected smart buildings apply communication standards that were initially not designed to work over the Internet Protocol; neither in IPv4 nor in IPv6. Instead of the previously separated building area networks, which can be attacked only locally, buildings now face attacks from the Internet as any other Internet of Things (IoT) device does. While many IoT devices are, however, designed with security in mind, building automation protocols still rely on legacy communication standards that, in many cases, did not foresee any mentionable security features. As a result, smart buildings can be easily attacked over the Internet.

Several academic contributions to improve the security of existing building automation protocols were developed under the umbrella of the Security in Building Automation project at TU Vienna.^d Within the last several years, vendors became aware of this problem and started to clearly improve the quality of communication protocol security specifications such as BACnet or KNX. From the perspective of a security researcher, this action takes place rather late and would have been needed 10 years ago but the building automation industry develops rather slowly. Building automation technology is designed to last for decades and is therefore highly robust against hardware failures. What is missing is the reliability against

several Internet-based attacks that were not foreseen when the particular protocols were developed.

Impact of Attacks. Second, consequences of attacks on smart buildings are not fully understood. These attacks do not target building automation equipment directly but instead focus on the physical environment of a building and its surrounding area. We need to differentiate between active and passive scenarios based on the physical capabilities of a building. In an active scenario, adversaries can influence the building's operation and with it the business process of an organization. For example, if legitimate physical access to a building is prevented by an access control system, a factory may not operate and if the fire alarm in an airport terminal is activated, the necessary evacuation causes all passenger transfers to stop. Actual "real-world" consequences of attacks are not addressed by current research and remain mostly unclear.

For an example of a passive scenario, consider the smart home. Using sensor values, a surveillance of inhabitants is feasible by either directly requesting sensor data such as presence sensors or information about the energy consumption of the home over the Internet or by exploiting side channels.³ Being connected with hospitals and wearable devices for elderly care, future smart buildings will transfer highly sensitive vital signs that potentially can be sold to health insurance companies on the black market. Such a surveillance

In recent years, vendors, due to market demand, started to connect their automated buildings to the Internet to provide new services.

scenario could, in a worst-case situation, lead to smart building botnets performing mass surveillance.⁵ Various additional attacks may still be uncovered and a systematic analysis of potential attacks outside of the technical level but on the level of organizational processes is necessary to envisage a full spectrum of threats.

Long-term Software Deployment.

Third, building automation equipment must be designed in a way that its integrated software will either be formally proven or it becomes 'patchable' in order to keep it secure on a long-term basis. A problem in this regard is the limited computing power and memory of the related embedded systems, especially for legacy devices.¹ Legacy devices are a significant problem in building automation: while it is easy to claim the building automation industry should implement more security features in their systems, it must be emphasized that automation equipment may last for decades in the same building. If we integrate state-of-the-art cryptography in a component for a building, we still need to ensure the component can handle the necessary state-of-the-art cryptography in a few decades to keep the building secure. A first step toward increased long-term security would be to support awareness for outdated software components. Therefore, building area networks could be scanned regularly by security software to provide information about obsolete software to building management interfaces.

User-Oriented Software Design.

Fourth, even if a long-term patchability will be realized technically, it must be designed in a way end users will actually use it. Otherwise, people will not patch their smart buildings for the same reason most users do not typically encrypt their private email. In general, innovative solutions to foster security awareness and to provide means to improve the security of smart buildings are required that are tailored to the different actors. Therefore, such solutions must consider the technical abilities and motivation of these actors, whether they are professional operators, owners, employees, or inhabitants, including the handicapped and the elderly.

d <https://www.auto.tuwien.ac.at/seBAS/>

For example, the Facility Using Smart Secured Energy & Information Technology (FUSE-IT) project develops a user-friendly security dashboard for building management. The dashboard will display potentially relevant information based on the actual requirements of its users.^e

Insecure Network Stacks. Fifth, another step toward more secure legacy systems is to address insecure network stack implementations. In the building automation industry, network stacks for complex protocols are, at least in some known cases, implemented by a single developer or very few developers without a software engineering or information security-related education. For this reason, the resulting software may be vulnerable to rather fundamental attacks. One option is to introduce a ‘normalization’ into the network environment of building automation systems. In the ongoing Building Automation Reliable Network Infrastructure (BARNI) project, funded by the German Ministry of Education and Research, we focus on this normalization aspect. By filtering network traffic and ensuring its conformity to security standards, we can refuse malformed network packets from reaching devices with insecure network stacks,² preventing these devices from failures. Another option is to perform penetration testing of network stacks, including traffic fuzzing. The penetration testing results can be used afterward for hardening the network stack.

Access to Standards. Sixth, a non-technical but momentous step toward better security of smart buildings is to provide the academic community with free access to related standards documents. Even some of the most important standards are only accessible under restrictions, for example, for payment. Providing students access to these standards for their projects and theses remains a challenge experienced multiple times within the last five years. The building automation community can learn from the Internet community, which publishes freely available Internet standard documents (even in early forms such as draft standards). Freely available stan-

People will not patch their smart buildings for the same reason most users do not typically encrypt their private email.

dards for building automation systems would allow the analysis of specifications by many scholars in the field of information security that currently have no access to these documents.

Conclusion

The introduction of improved security into smart buildings requires the collaboration of academia and industry. This collaboration already started for selected sub-topics but an interdisciplinary evaluation of potential threats in smart buildings should be a next step to conclude the full spectrum of potential risks. In a following phase of joint work, security technology must be developed for both new and legacy buildings, tailored for the different involved actors to address this threat landscape. **C**

References

1. Granzer, W., Praus, F. and Kastner, W. Security in building automation systems. *IEEE Transactions on Industrial Electronics* 57, 11 (Nov. 2010), 3622–3630.
2. Kaur, J., Tonejc, J., Wendzel, S. and Meier, M. Securing BACnet's pitfalls. In *Proceedings of the 30th International Information Security and Privacy Conference (IFIP Sec)*, AICT 445, Springer, (2015), 616–629.
3. Krüger, F., Kasparick, M., Mundt, T., and Kirste, T. Where are my colleagues and why? Tracking multiple persons in indoor environments. In *Proceedings of the International Conference on Intelligent Environments*. IEEE Computer Society (2014), 190–197.
4. Soucek, S. and Zucker, G. Current developments and challenges in building automation. *e & i (Elektrotechnik und Informationstechnik)* 129, 4 (Apr. 2012), Springer, 278–285.
5. Wendzel, S., Zwanger, V., Meier, M. and Szlósarczyk, S. Envisioning smart building botnets. In *Proceedings of GI Sicherheit*, LNI 228 (2014), 319–329.
6. Wong, A.C.W. and So, A.T.P. Building automation in the 21st century. In *Proceedings of the 4th International Conference on Advances in Power System Control, Operation, Management (APSCOM'97)*, vol. 2 (Nov. 1997), IET, 819–824.

Steffen Wendzel (steffen.wendzel@fkie.fraunhofer.de) leads a research team on secure building automation at the Fraunhofer Institute for Communication, Information Processing, and Ergonomics (FKIE), Bonn, Germany.

Copyright held by author.

Calendar of Events

May 7–8

MIG '15: Motion in Games, Lisbon, Portugal, Sponsored: ACM/SIG, Contact: Catherine Pelachaud, Email: catherine.pelachaud@telecom-paristech.fr

May 7–9

Expressive '16: The Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering, Lisbon, Portugal, Contact: Manuel Joao Fonseca, Email: mjfonseca@ciencias.ulisboa.pt

May 7–12

CHI'16: CHI Conference on Human Factors in Computing Systems, San Jose, CA, Sponsored: ACM/SIG, Contact: Allison Druin, Email: allisond@umiacs.umd.edu

May 14–22

ICSE '16: 38th International Conference on Software Engineering, Austin, TX, Contact: Laura Dillon, Email: ldillon@cse.msu.edu

May 15–16

MOBILESoft '16: International Conference on Mobile Software Engineering and Systems, Austin, TX, Contact: Aharon Abadi, Email: aharon@il.ibm.com

May 15–18

SIGSIM-PADS '16: SIGSIM Principles of Advanced Discrete Simulation, Banff, AB, Canada, Sponsored: ACM/SIG, Contact: Richard M. Fujimoto, Email: fujimoto@cc.gatech.edu

May 16–19

CCGrid '16: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Cartagena, Colombia, Email: balaji@mcsl.anl.gov

^e <http://www.itea2-fuse-it.com>

Lessons learned from three container-management systems over a decade.

BY BRENDAN BURNS, BRIAN GRANT, DAVID OPPENHEIMER,
ERIC BREWER, AND JOHN WILKES

Borg, Omega, and Kubernetes

THOUGH WIDESPREAD INTEREST in software containers is a relatively recent phenomenon, at Google we have been managing Linux containers at scale for more than 10 years and built three different container-management systems in that time. Each system is heavily influenced by its predecessors, even though they were developed for different reasons. This article describes the lessons we've learned from developing and operating them.

The first unified container-management system developed at Google was the system we internally call Borg.⁷ It was built to manage both long-running services and batch jobs, which had previously been handled by two separate systems: Babysitter and the Global Work Queue. The latter's architecture strongly influenced Borg, but was focused on batch jobs; both predated Linux control groups. Borg shares machines between these two types of applications as a way of

increasing resource utilization and thereby reducing costs. Such sharing was possible because container support in the Linux kernel was becoming available (indeed, Google contributed much of the container code to the Linux kernel), which enabled better isolation between latency-sensitive user-facing services and CPU-hungry batch processes.

As more applications were developed to run on top of Borg, our application and infrastructure teams developed a broad ecosystem of tools and services for it. These systems provided mechanisms for configuring and updating jobs; predicting resource requirements; dynamically pushing configuration files to running jobs; service discovery and load balancing; auto-scaling; machine-lifecycle management; quota management; and much more. The development of this ecosystem was driven by the needs of different teams inside Google, and the result was a somewhat heterogeneous, ad hoc collection of systems that Borg's users had to configure and interact with, using several different configuration languages and processes. Borg remains the primary container-management system within Google because of its scale, breadth of features, and extreme robustness.

Omega,⁶ an offspring of Borg, was driven by a desire to improve the software engineering of the Borg ecosystem. It applied many of the patterns that had proved successful in Borg, but was built from the ground up to have a more consistent, principled architecture. Omega stored the state of the cluster in a centralized Paxos-based transaction-oriented store accessed by the different parts of the cluster control plane (such as schedulers), using optimistic concurrency control to handle the occasional conflicts. This decoupling allowed the Borgmaster's functionality to be broken into separate components that acted as peers, rather than funneling every change through a monolithic, centralized master. Many of Omega's innovations



(including multiple schedulers) have since been folded into Borg.

The third container-management system developed at Google was Kubernetes.⁴ It was conceived of and developed in a world where external developers were becoming interested in Linux containers, and Google had developed a growing business selling public-cloud infrastructure. Kubernetes is open source—a contrast to Borg and Omega, which were developed as purely Google-internal systems. Like Omega, Kubernetes has at its core a shared persistent store, with components watching for changes to relevant objects. In contrast to Omega, which exposes the store directly to trusted control-plane components, state in Kubernetes is accessed exclusively through a domain-specific REST API that applies higher-level versioning, validation, semantics, and policy, in support of a more diverse array of clients. More importantly, Kubernetes was developed with a stronger focus on the experience of developers writing applications that run in a cluster: its main design goal is to make it easy to deploy and manage complex distributed systems, while still benefiting from the improved utilization that containers enable.

This article describes some of the knowledge gained and lessons learned during Google's journey from Borg to Kubernetes.

Containers

Historically, the first containers just provided isolation of the root file system (via chroot), with FreeBSD jails extending this to additional namespaces such as process IDs. Solaris subsequently pioneered and explored many enhancements. Linux control groups (cgroups) adopted many of these ideas, and development in this area continues today.

The resource isolation provided by containers has enabled Google to drive utilization significantly higher than industry norms. For example, Borg uses containers to co-locate batch jobs with latency-sensitive, user-facing jobs on the same physical machines. The user-facing jobs reserve more resources than they usually need—allowing them to handle load spikes and fail-over—and these mostly unused resources can be



Over time it became clear that the benefits of containerization go beyond merely enabling higher levels of utilization.



reclaimed to run batch jobs. Containers provide the resource-management tools that make this possible, as well as robust kernel-level resource isolation to prevent the processes from interfering with one another. We achieved this by enhancing Linux containers concurrently with Borg's development. The isolation is not perfect, though: containers cannot prevent interference in resources that the operating-system kernel does not manage, such as level 3 processor caches and memory bandwidth, and containers need to be supported by an additional security layer (such as virtual machines) to protect against the kinds of malicious actors found in the cloud.

A modern container is more than just an isolation mechanism: it also includes an *image*—the files that make up the application that runs inside the container. Within Google, MPM (Midas Package Manager) is used to build and deploy container images. The same symbiotic relationship between the isolation mechanism and MPM packages can be found between the Docker daemon and the Docker image registry. In the remainder of this article we use the word *container* to encompass both of these aspects: the runtime isolation and the image.

Application-Oriented Infrastructure

Over time it became clear that the benefits of containerization go beyond merely enabling higher levels of utilization. Containerization transforms the data center from being machine-oriented to being application-oriented. This section discusses two examples:

- Containers encapsulate the application environment, abstracting away many details of machines and operating systems from the application developer and the deployment infrastructure.

- Because well-designed containers and container images are scoped to a single application, managing containers means managing applications rather than machines. This shift of management APIs from machine-oriented to application-oriented dramatically improves application deployment and introspection.

Application environment. The original purpose of the cgroup, chroot, and namespace facilities in the kernel was to protect applications from noisy, nos-

ey, and messy neighbors. Combining these with container images created an abstraction that also isolates applications from the (heterogeneous) operating systems on which they run. This decoupling of image and OS makes it possible to provide the same deployment environment in both development and production, which, in turn, improves deployment reliability and speeds up development by reducing inconsistencies and friction.

The key to making this abstraction work is having a *hermetic* container image that can encapsulate almost all of an application's dependencies into a package that can be deployed into the container. If this is done correctly, the only local external dependencies will be on the Linux kernel system-call interface. While this limited interface dramatically improves the portability of images, it is not perfect: applications can still be exposed to churn in the OS interface, particularly in the wide surface area exposed by socket options, `/proc`, and arguments to `ioctl` calls. Our hope is that ongoing efforts such as the Open Container Initiative (<https://www.opencontainers.org/>) will further clarify the surface area of the container abstraction.

Nonetheless, the isolation and dependency minimization provided by containers have proved quite effective at Google, and the container has become the sole runnable entity supported by the Google infrastructure. One consequence is that Google has only a small number of OS versions deployed across its entire fleet of machines at any one time, and it needs only a small staff of people to maintain them and push out new versions.

There are many ways to achieve these hermetic images. In Borg, program binaries are statically linked at build time to known-good library versions hosted in the company-wide repository.⁵ Even so, the Borg container image is not quite as airtight as it could have been: applications share a so-called *base image* that is installed once on the machine rather than being packaged in each container. This base image contains utilities such as `tar` and the `libc` library, so upgrades to the base image can affect running applications and have occasionally been a significant source of trouble.

More modern container image formats such as Docker and ACI harden this abstraction further and get closer to the hermetic ideal by eliminating implicit host OS dependencies and requiring an explicit user command to share image data between containers.

Containers as the unit of management. Building management APIs around containers rather than machines shifts the “primary key” of the data center from machine to application. This has many benefits: [1] it relieves application developers and operations teams from worrying about specific details of machines and operating systems; [2] it provides the infrastructure team flexibility to roll out new hardware and upgrade operating systems with minimal impact on running applications and their developers; [3] it ties telemetry collected by the management system (for example, metrics such as CPU and memory usage) to applications rather than machines, which dramatically improves application monitoring and introspection, especially when scale-up, machine failures, or maintenance cause application instances to move.

Containers provide convenient points to register generic APIs that enable the flow of information between the management system and an application without either knowing much about the particulars of the other's implementation. In Borg, this API is a series of HTTP endpoints attached to each container. For example, the `/healthz` endpoint reports application health to the orchestrator. When an unhealthy application is detected, it is automatically terminated and restarted. This self-healing is a key building block for reliable distributed systems. (Kubernetes offers similar functionality; the health check uses a user-specified HTTP endpoint or `exec` command that runs inside the container.)

Additional information can be provided by or for containers and displayed in various user interfaces. For example, Borg applications can provide a simple text status message that can be updated dynamically, and Kubernetes provides key-value *annotations* stored in each object's metadata that can be used to communicate application structure. Such annotations can be set by the container itself or oth-

er actors in the management system (for example, the process rolling out an updated version of the container).

In the other direction, the container-management system can communicate information into the container such as resource limits, container metadata for propagation to logging and monitoring (for example, user name, job name, identity), and notices that provide graceful-termination warnings in advance of node maintenance.

Containers can also provide application-oriented monitoring in other ways: for example, Linux kernel cgroups provide resource-utilization data about the application, and these can be extended with custom metrics exported using HTTP APIs, as described earlier. This data enables the development of generic tools like an auto-scaler or `cAdvisor`³ that can record and use metrics without understanding the specifics of each application. Because the container is the application, there is no need to (de)multiplex signals from multiple applications running inside a physical or virtual machine. This is simpler, more robust, and permits finer-grained reporting and control of metrics and logs. Compare this to having to `ssh` into a machine to run `top`. Though it is possible for developers to `ssh` into their containers, they rarely need to.

Monitoring is just one example. The application-oriented shift has ripple effects throughout the management infrastructure. Our load balancers don't balance traffic across machines; they balance across application instances. Logs are keyed by application, not machine, so they can easily be collected and aggregated across instances without pollution from multiple applications or system operations. We can detect application failures and more readily ascribe failure causes without having to disentangle them from machine-level signals. Fundamentally, because the identity of an instance being managed by the container manager lines up exactly with the identity of the instance expected by the application developer, it is easier to build, manage, and debug applications.

Finally, although so far we have focused on applications being 1:1 with

containers, in reality we use nested containers that are co-scheduled on the same machine: the outermost one provides a pool of resources; the inner ones provide deployment isolation. In Borg, the outermost container is called a resource allocation, or *alloc*; in Kubernetes, it is called a *pod*. Borg also allows top-level application containers to run outside allocs; this has been a source of much inconvenience, so Kubernetes regularizes things and always runs an application container inside a top-level pod, even if the pod contains a single container.

A common use pattern is for a pod to hold an instance of a complex application. The major part of the application sits in one of the child containers, and other containers run supporting functions such as log rotation or click-log offloading to a distributed file system. Compared to combining the functionality into a single binary, this makes it easy to have different teams develop the distinct pieces of functionality, and it improves robustness (the offloading continues even if the main application gets wedged), composability (it is easy to add a new small support service, because it operates in the private execution environment provided by its own container), and fine-grained resource isolation (each runs in its own resources, so the logging system cannot starve the main app, or vice versa).

Orchestration is the beginning, not the end. The original Borg system made it possible to run disparate workloads on shared machines to improve resource utilization. The rapid evolution of support services in the Borg ecosystem, however, showed that container management *per se* was just the beginning of an environment for developing and managing reliable distributed systems. Many different systems have been built in, on, and around Borg to improve upon the basic container-management services that Borg provided. The following partial list gives an idea of their range and variety:

- ▶ Naming and service discovery (the Borg Name Service, or BNS).
- ▶ Master election, using Chubby.²
- ▶ Application-aware load balancing.
- ▶ Horizontal (number of instances) and vertical (size of an instance) auto-scaling.

▶ Rollout tools that manage the careful deployment of new binaries and configuration data.

▶ Workflow tools (for example, to allow running multi-job analysis pipelines with interdependencies between the stages).

▶ Monitoring tools to gather information about containers, aggregate it, present it on dashboards, and use it to trigger alerts.

These services were built organically to solve problems that application teams experienced. The successful ones were picked up, adopted widely, and made other developers' lives easier. Unfortunately, these tools typically picked idiosyncratic APIs, conventions (such as file locations), and depth of Borg integration. An undesired side effect was to increase the complexity of deploying applications in the Borg ecosystem.

Kubernetes attempts to avert this increased complexity by adopting a consistent approach to its APIs. For example, every Kubernetes object has three basic fields in its description: `ObjectMetadata`, `Specification` (or `Spec`), and `Status`.

The `ObjectMetadata` is the same for all objects in the system; it contains information such as the object's name, UID (unique identifier), an object version number (for optimistic concurrency control), and labels (key-value pairs, see below). The contents of `Spec` and `Status` vary by object type, but their concept does not: `Spec` is used to describe the *desired* state of the object, whereas `Status` provides read-only information about the *current* state of the object.

This uniform API provides many benefits. Learning the system is simpler: similar information applies to all objects, and writing generic tools that work across all objects is simpler, which in turn enables the development of a consistent user experience. Learning from Borg and Omega, Kubernetes is built from a set of composable building blocks that can readily be extended by its users. A common API and object-metadata structure makes that much easier. For example, the pod API is usable by people, internal Kubernetes components, and external automation tools. To further this consistency, Kubernetes is being extended to enable users to add their

own APIs dynamically, alongside the core Kubernetes functionality.

Consistency is also achieved via decoupling in the Kubernetes API. Separation of concerns between API components means higher-level services all share the same common basic building blocks. A good example of this is the separation between the Kubernetes replica controller and its horizontal auto-scaling system. A replication controller ensures the existence of the desired number of pods for a given role (for example, "front end"). The auto-scaler, in turn, relies on this capability and simply adjusts the desired number of pods, without worrying about how those pods are created or deleted. The autoscaler implementation can focus on demand and usage predictions, and ignore the details of how to implement its decisions.

Decoupling ensures multiple related but different components share a similar look and feel. For example, Kubernetes has three different forms of replicated pods:

▶ `ReplicationController`: run-forever replicated containers (for example, Web servers).

▶ `DaemonSet`: ensure a single instance on each node in the cluster (for example, logging agents).

▶ `Job`: a run-to-completion controller that knows how to run a (possibly parallelized) batch job from start to finish.

Regardless of the differences in policy, all three of these controllers rely on the common pod object to specify the containers they wish to run.

Consistency is also achieved through common design patterns for different Kubernetes components. The idea of a *reconciliation controller loop* is shared throughout Borg, Omega, and Kubernetes to improve the resiliency of a system: it compares a desired state (for example, how many pods should match a label-selector query) against the observed state (the number of such pods that it can find), and takes actions to converge the observed and desired states. Because all action is based on observation rather than a state diagram, reconciliation loops are robust to failures and perturbations: when a controller fails or restarts it simply picks up where it left off.

The design of Kubernetes as a com-

bination of microservices and small control loops is an example of control through *choreography*—achieving a desired emergent behavior by combining the effects of separate, autonomous entities that collaborate. This is a conscious design choice in contrast to a centralized *orchestration system*, which may be easier to construct at first but tends to become brittle and rigid over time, especially in the presence of unanticipated errors or state changes.

Things to Avoid

While developing these systems we have learned almost as many things *not* to do as ideas that are worth doing. We present some of them here in the hopes that others can focus on making new mistakes, rather than repeating ours.

Don't make the container system manage port numbers. All containers running on a Borg machine share the host's IP address, so Borg assigns the containers unique port numbers as part of the scheduling process. A container will get a new port number when it moves to a new machine and (sometimes) when it is restarted on the same machine. This means traditional networking services such as the DNS (Domain Name System) have to be replaced by home-brew versions; service clients do not know the port number assigned to the service *a priori* and have to be told; port numbers cannot be embedded in URLs, requiring name-based redirection mechanisms; and tools that rely on simple IP addresses need to be rewritten to handle IP:port pairs.

Learning from our experiences with Borg, we decided Kubernetes would allocate an IP address per pod, thus aligning network identity (IP address) with application identity. This makes it much easier to run off-the-shelf software on Kubernetes: applications are free to use static well-known ports (for example, 80 for HTTP traffic), and existing, familiar tools can be used for things like network segmentation, bandwidth throttling, and management. All of the popular cloud platforms provide networking underlays that enable IP-per-pod; on bare metal, one can use a Software Defined Network (SDN) overlay or configure L3 routing to handle multiple IPs per machine.



The design of Kubernetes as a combination of microservices and small control loops is an example of control through choreography.



Don't just number containers; give them labels. If you allow users to create containers easily, they tend to create lots of them, and soon need a way to group and organize them. Borg provides *jobs* to group identical *tasks* (its name for containers). A job is a compact vector of one or more identical tasks, indexed sequentially from zero. This provides a lot of power and is simple and straightforward, but we came to regret its rigidity over time. For example, when a task dies and has to be restarted on another machine, the same slot in the task vector has to do double duty: to identify the new copy and to point to the old one in case it needs to be debugged. When tasks in the middle of the vector exit, the vector ends up with holes. The vector makes it very hard to support jobs that span multiple clusters in a layer above Borg. There are also insidious, unexpected interactions between Borg's job-update semantics (which typically restarts tasks in index order when doing rolling upgrades) and an application's use of the task index (for example, to do sharding or partitioning of a dataset across the tasks): if the application uses range sharding based on the task index, Borg's restart policy can cause data unavailability, as it takes down adjacent tasks. Borg also provides no easy way to add application-relevant metadata to a job, such as role (for example, "frontend"), or rollout status (for example, "canary"), so people encode this information into job names that they decode using regular expressions.

In contrast, Kubernetes primarily uses *labels* to identify groups of containers. A label is a key/value pair that contains information that helps identify the object. A pod might have the labels `role=frontend` and `stage=production`, indicating this container is serving as a production front-end instance. Labels can be dynamically added, removed, and modified by either automated tools or users, and different teams can manage their own labels largely independently. Sets of objects are defined by *label selectors* (for example, `stage==production && role==frontend`). Sets can overlap, and an object can be in multiple sets, so labels are inherently more flexible than explicit lists of objects or simple static properties. Because a set

is defined by a dynamic query, a new one can be created at any time. Label selectors are *the* grouping mechanism in Kubernetes, and define the scope of all management operations that can span multiple entities.

Even in those circumstances where knowing the identity of a task in a set is helpful (for example, for static role assignment and work-partitioning or sharding), appropriate per-pod labels can be used to reproduce the effect of task indexes, though it is the responsibility of the application (or some other management system external to Kubernetes) to provide such labeling. Labels and label selectors provide a general mechanism that gives the best of both worlds.

Be careful with ownership. In Borg, tasks do not exist independently from jobs. Creating a job creates its tasks; those tasks are forever associated with that particular job, and deleting the job deletes the tasks. This is convenient, but it has a major drawback: because there is only one grouping mechanism, it needs to handle all use cases. For example, a job has to store parameters that make sense only for service or batch jobs but not both, and users must develop workarounds when the job abstraction doesn't handle a use case (for example, a `DaemonSet` that replicates a single pod to all nodes in the cluster).

In Kubernetes, pod-lifecycle management components such as replication controllers determine which pods they are responsible for using label selectors, so multiple controllers might think they have jurisdiction over a single pod. It is important to prevent such conflicts through appropriate configuration choices. But the flexibility of labels has compensating advantages—for example, the separation of controllers and pods means it is possible to “orphan” and “adopt” containers. Consider a load-balanced service that uses a label selector to identify the set of pods to send traffic to. If one of these pods starts misbehaving, that pod can be quarantined from serving requests by removing one or more of the labels that cause it to be targeted by the Kubernetes *service* load balancer. The pod is no longer serving traffic, but it will remain up and can be debugged *in situ*. In the meantime, the



If you allow users to create containers easily, they tend to create lots of them, and soon need a way to group and organize them.



replication controller managing the pods that implement the service automatically creates a replacement pod for the misbehaving one.

Don't expose raw state. A key difference between Borg, Omega, and Kubernetes is in their API architectures. The Borgmaster is a monolithic component that knows the semantics of *every* API operation. It contains the cluster management logic such as the state machines for jobs, tasks, and machines; and it runs the Paxos-based replicated storage system used to record the master's state. In contrast, Omega has no centralized component except the store, which simply holds passive state information and enforces optimistic concurrency control: all logic and semantics are pushed into the clients of the store, which directly read and write the store contents. In practice, every Omega component uses the same client-side library for the store, which does packing/unpacking of data structures, retries, and enforces semantic consistency.

Kubernetes picks a middle ground that provides the flexibility and scalability of Omega's componentized architecture while enforcing system-wide invariants, policies, and data transformations. It does this by forcing all store accesses through a centralized API server that hides the details of the store implementation and provides services for object validation, defaulting, and versioning. As in Omega, the client components are decoupled from one another and can evolve or be replaced independently (which is especially important in the open source environment), but the centralization makes it easy to enforce common semantics, invariants, and policies.

Some Open, Hard Problems

Even with years of container-management experience, we feel there are a number of problems that we still don't have answers for. This section describes a couple of particularly knotty ones, in the hope of fostering discussion and solutions.

Configuration. Of all the problems we have confronted, the ones over which the most brainpower, ink, and code have been spilled are related to managing *configurations*—the set of values supplied to applications, rather than hard-coded into them. In

truth, we could have devoted this entire article to the subject and still have had more to say. What follows are a few highlights.

First, application configuration becomes the catch-all location for implementing all of the things the container-management system doesn't (yet) do. Over the history of Borg this has included:

- ▶ Boilerplate reduction (for example, defaulting task-restart policies appropriate to the workload, such as service or batch jobs).
- ▶ Adjusting and validating application parameters and command-line flags.
- ▶ Implementing workarounds for missing API abstractions such as package (image) management.
- ▶ Libraries of configuration templates for applications.
- ▶ Release-management tools.
- ▶ Image version specification.

To cope with these kinds of requirements, configuration-management systems tend to invent a domain-specific configuration language that (eventually) becomes Turing complete, starting from the desire to perform computation on the data in the configuration (for example, to adjust the amount of memory to give a server as a function of the number of shards in the service). The result is the kind of inscrutable “configuration is code” that people were trying to avoid by eliminating hard-coded parameters in the application's source code. It doesn't reduce operational complexity or make the configurations easier to debug or change; it just moves the computations from a real programming language to a domain-specific one, which typically has weaker development tools such as debuggers and unit test frameworks.

We believe the most effective approach is to accept this need, embrace the inevitability of programmatic configuration, and maintain a clean separation between computation and data. The language to represent the data should be a simple, data-only format such as JSON or YAML, and programmatic modification of this data should be done in a real programming language, where there are well-understood semantics, as well as good tooling. Interestingly, this same separation of computation and data can be seen in front-end development

with frameworks such as Angular that maintain a crisp separation between the worlds of markup (data) and JavaScript (computation).

Dependency management. Standing up a service typically also means standing up a series of related services (monitoring, storage, CI/CD, etc). If an application has dependencies on other applications, wouldn't it be nice if those dependencies (and any transitive dependencies they may have) were automatically instantiated by the cluster-management system?

To complicate things, instantiating the dependencies is rarely as simple as just starting a new copy—for example, it may require registering as a consumer of an existing service (for example, Bigtable as a service) and passing authentication, authorization, and billing information across those transitive dependencies. Almost no system, however, captures, maintains, or exposes this kind of dependency information, so automating even common cases at the infrastructure level is nearly impossible. Turning up a new application remains complicated for the user, making it more difficult for developers to build new services, and often results in the most recent best practices not being followed, which affects the reliability of the resulting service.

A standard problem is that is hard to keep dependency information up to date if it is provided manually, and at the same time attempts to determine it automatically (for example, by tracing accesses) fail to capture the semantic information needed to understand the result. (Did that access have to go to *that* instance, or would any instance have sufficed?) One possible way to make progress is to require that an application enumerate the services on which it depends, and have the infrastructure refuse to allow access to any others. (We do this for compiler imports in our build system.¹) The incentive would be enabling the infrastructure to do useful things in return, such as automatic setup, authentication, and connectivity.

Unfortunately, the perceived complexity of systems that express, analyze, and use system dependencies has been too high, and so they haven't yet been added to a mainstream container-

management system. We still hope Kubernetes might be a platform on which such tools can be built, but doing so remains an open challenge.

Conclusion

A decade's worth of experience building container-management systems has taught us much, and we have embedded many of those lessons into Kubernetes, Google's most recent container-management system. Its goals are to build on the capabilities of containers to provide significant gains in programmer productivity and ease of both manual and automated system management. We hope you'll join us in extending and improving it. □

Related articles on queue.acm.org

Storage Virtualization Gets Smart

Kostadis Roussos

<http://queue.acm.org/detail.cfm?id=1317404>

Unikernels: Rise of the Virtual Library Operating System

Anil Madhavapeddy and David J. Scott

<http://queue.acm.org/detail.cfm?id=2566628>

References

1. Bazel. [fast, correct]—choose two; <http://bazel.io>.
2. Burrows, M. The Chubby lock service for loosely coupled distributed systems. In *Proceedings from the Symposium on Operating System Design and Implementation (OSDI)*. Seattle, WA, 2006.
3. cAdvisor; <https://github.com/google/cadvisor>.
4. Kubernetes; <http://kubernetes.io/>.
5. Metz, C. Google is 2 billion lines of code—and it's all in one place. *Wired* (Sept. 2015); <http://www.wired.com/2015/09/google-2-billion-lines-codeand-one-place/>.
6. Schwarzkopf, M., Konwinski, A., Abd-el-Malek and M., Wilkes, J. Omega: Flexible, scalable schedulers for large compute clusters. In *Proceedings from the European Conference on Computer Systems (EuroSys)*. Prague, Czech Republic, 2013.
7. Verma, A., Pedrosa, L., Korupolu, M. R., Oppenheimer, D., Tune, E., Wilkes, J. Large-scale cluster management at Google with Borg. In *Proceedings from the European Conference on Computer Systems (EuroSys)*. Bordeaux, France, 2015.

Brendan Burns (@brendandburns) is a software engineer at Google, where he co-founded the Kubernetes project. Prior to working on Kubernetes and cloud, he worked on low-latency indexing for Google's web-search infrastructure.

Brian Grant is a software engineer at Google. He was previously a technical lead of Borg and founder of the Omega project and is now design lead of Kubernetes.

David Oppenheimer is a software engineer at Google and a tech lead on the Kubernetes project. He was a tech lead on Google's Borg and Omega cluster-management systems prior to Kubernetes.

Eric Brewer is VP Infrastructure at Google and a professor at UC Berkeley, where he pioneered scalable servers and elastic infrastructure.

John Wilkes has been working on cluster management and infrastructure services at Google since 2008. Previously, he spent time at HP Labs. His research interests include technologies that allow systems to manage themselves.

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

Be someone who makes everyone else better.

BY KATE MATSUDAIRA

Delegation as Art

WHEN I STARTED my career as a junior engineer, I could not wait to be senior. I would regularly review our promotion guidelines and assess my progress and contributions against them. Of course, at the time I did not really understand what being senior meant.

Being a senior engineer means having strong technical skills, the ability to communicate well and navigate ambiguous situations, and most important of all, the ability to grow and lead other people.

Leadership is not just for managers anymore.

I have worked with many genuinely talented engineers throughout my career. I have noticed that truly great engineers bring forth the capabilities of everyone around them. Having them on the team makes everyone brighter, more productive, and more motivated.

The difference between being an engineer and being a senior engineer often has to do with the ability to mentor and grow your teammates. Mentoring is all about *teaching* and *delegation*.

Adding senior to your title means you have the most experience (whether that is industry experience working with a specific set of technologies, or methodologies,

or simply legacy knowledge of the systems you are using). And because you are senior, part of your job is imparting that knowledge onto others.

When done well, it is an amazing thing:

- ▶ All of a sudden you have time to work on new project XYZ instead of being burdened with legacy operations, support, and fixes.

- ▶ You get to watch your colleagues grow and learn, taking on more responsibility and building their skills.

- ▶ Everyone is more productive and hard working because they are able to learn from past mistakes and take on new challenges.

- ▶ Woo!

Taking on the role of teacher can also be very difficult:

- ▶ When you were the only one who





knew system ABC, you were so important. Everyone recognized you and came to you for help—and it felt good to be needed.

► It can be fun to work on stuff that you are the expert on—it is nice to be able to tackle a task you know how to do (blindfolded or with one hand tied behind your back).

► “If I hand this over to someone else, I am worried they are going to screw up all of the great work I have done.”

► When it comes to delegating, it seems like it can take much longer to explain how to do something to someone else than it would take you to get the work done yourself—and who does not love to be efficient?

Even though you know it is beneficial, delegating is difficult. You not

only have to be able to trust others to get the work done, but you also must know how much information to give them that will enable them to make progress. This can be really difficult to do on gnarly technical problems. How do you delegate tasks and teach others about something that seems very un-delegatable?

I have seen this many times, and from all of my experience I have come up with some suggestions.

Change your mind-set. You have heard that old saying, “Give a man a fish and you feed him for a day; teach a man to fish and you feed him for a lifetime.” Well, it applies in your career, too. If you think taking the time to explain something is not worth it, then you will never be able to teach anyone else. Sure, it takes more time than do-

ing the task yourself, but that time is valuable to the person you are teaching. You have to focus on growth and education, not necessarily efficiency.

Start small. This is the key to successful mentoring and delegating. You want to start with something that is easy to do—for example, fixing a small bug, writing test cases, or adding documentation. When you start with something very tractable, it gives the other person a chance to get his or her feet wet and a moment to peek inside the inner workings without getting overwhelmed. When you want to train someone on something new, try to find small things that are not urgent—iterative improvements that would be an easy introduction into your realm of responsibility; repeatable things.

Lead them to figure it out. It is easy

to give people answers when they ask you questions. Next time this happens, resist the urge to tell them how to solve the problem. Instead ask them questions like:

- ▶ What have you tried so far?
- ▶ What have you considered trying?
- ▶ What did you research? Did you find anything?
- ▶ Have you tried looking up _____? (the blank representing what you would look for to find the answer).

You want to show them the steps you would take and lead them through your thought process—how you would solve the problem—instead of simply providing the solution.

Help them come up with a plan. As you give people more responsibility (for example, building a new feature), do not just let them go out on their own. Instead have them draft up a plan and bring it back to you. Go over their approach together. Give them feedback (ideally, by asking good questions instead of just giving them answers). When you draft the idea together, you will feel more confident in their approach and they will be able to leverage your expertise and direction.

Set up some guardrails. Whenever someone takes on new responsibility in your team or organization it is good practice to set up some checks and balances to ensure things go smoothly (especially if you are handing over a mission-critical part of software or operations). Figure out what information you need, and how frequently, so you can feel confident things are where they should be. For example, you could set up daily status meetings to go over progress, biweekly code reviews to go over implementation, or weekly one-on-ones with other people on the project. Try to identify what information you need to feel good about the progress and then the best way to get that information without being overbearing or micromanaging the details.

Don't look for perfection. People learning something new are unlikely to get it right the first time. And when it comes to code, it seems like programmers are more like artists, with each individual interpreting a task a little bit differently. Make sure you are open to these differences and you accept

Being a good leader means allowing the people around you to be the experts in their domains.

that other people may not do things the same way you would. If you are really worried about quality, specify your definition of quality up front when you assign the task.

Create a team of leaders. Last year I read the book *Turn the Ship Around!: A True Story of Turning Followers into Leaders* by L. David Marquet (Portfolio, 2013) and loved it. One of the most substantial lessons I learned from the book was to adopt a leader:leader model instead of a leader:follower model. Being a good leader means allowing the people around you to be the experts in their domains. My favorite concept from Marquet's book that truly changed the way I lead was this: when someone on my team asks me a question, instead of giving them the answer, I ask them, "What do you think I would say?" Most of the time their answer aligns with my thinking, but when it does not, I realize they are missing some important information or context I have neglected to share. This transformed my team; they stopped asking me questions and instead started making recommendations and asking for permission. How can you encourage your teammates to take more ownership and lead their own arenas?

Learning to coach and delegate is not easy, but once you build these skills, you will become even more valuable to your team. You will not just be the expert; you will be someone who makes everyone better—and isn't that the type of person you would want to work with? 

Related articles on queue.acm.org

Best Practice BPM

Derek Miers

<http://queue.acm.org/detail.cfm?id=1122688>

Evolution of the Product Manager

Ellen Chisa

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

The Science of Managing Data Science

Kate Matsudaira

<http://queue.acm.org/detail.cfm?id=2767971>

Kate Matsudaira (katemats.com) is the founder of her own company, Popforms. Previously she worked at Microsoft and Amazon as well as startups like Decide, Moz, and Delve Networks.

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



The hub of software development.

BY IVAR JACOBSON, IAN SPENCE, AND BRIAN KERR

Use-Case 2.0

USE CASES HAVE been around for almost 30 years as a requirements approach and have been part of the inspiration for more recent techniques such as user stories. Now the inspiration has flown in the other direction. Use-Case 2.0 is the new generation of use-case-driven development—light, agile, and

lean—inspired by user stories and the agile methodologies Scrum and Kanban.

Use-Case 2.0 has all the popular values from the past—not just supporting requirements, but also architecture, design, test, and user experience—and it is instrumental in business modeling and software reuse.

Use-Case 2.0 has been inspired by user stories to assist with backlogs à la Scrum and one-piece flow with Kanban, with the introduction of an important new concept, the use-case slice.

This article makes the argument that use cases essentially include the techniques provided by user stories but offer significantly more for larger systems, larger teams, and more complex and demanding developments. They are as lightweight as user stories, but can also scale in a smooth and structured way to incorporate as much detail as needed. Most importantly,

they drive and connect many other aspects of software development.

Use cases were introduced at OOPS-LA 87,⁶ although they were not widely adopted until the publication of the 1992 book *Object-Oriented Software Engineering—A Use-Case-Driven Approach*.⁷ Since then many other authors have adopted parts of the idea, notably Alistair Cockburn² concerning requirements and Larry Constantine⁴ regarding designing for better user experiences. Use cases were adopted as a part of the standard UML (Unified Modeling Language¹), and its diagrams (the use case and the actor icons) are among the most widely used parts of the language. Many other books and papers have been written about use cases for all kinds of systems—not just for software, but also business, systems (such as embedded systems), and systems of systems. Focusing on today and the

Figure 1. Use cases are the hub of software development.

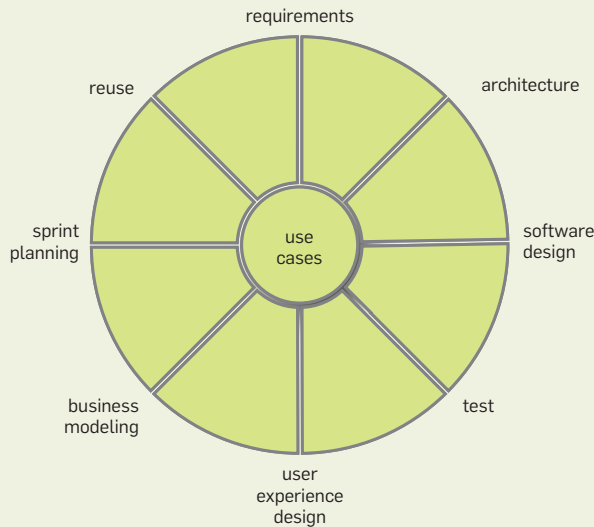


Figure 2. Use-case diagram for a simple telephone system.

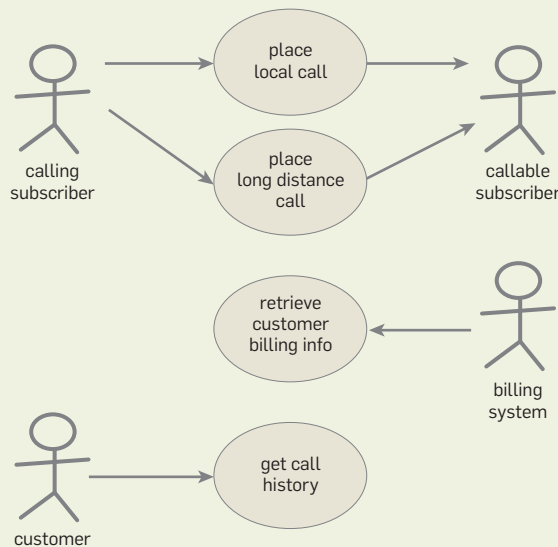


Figure 3. Structure of a use-case narrative.

basic flow

1. insert card
2. validate card
3. select cash withdrawal
4. select account
5. confirm availability of funds
6. return card
7. dispense cash

alternative flows

- A1 invalid card
- A2 non-standard amount
- A3 receipt required
- A4 insufficient funds in ATM
- A5 insufficient funds in account
- A6 would cause overdraft
- A7 card stuck
- A8 cash left behind
- etc.

future, the latest macro-trends, the Internet of Things (IoT), and Industrial Internet, have made use cases their choice.⁹

The use-case practice has evolved over the years, inspired by ideas from many different people, with the newer ideas incorporated into Use-Case 2.0. One new idea is slicing a use case into use-case slices.^{5,8} The idea of sizing these slices to become suitable backlog items and relating them to user stories is at the core of Use-Case 2.0.

Use cases can and should be used to drive software development. They do not prescribe how you should plan or manage your development work, or how you should design, develop, or test your system. They do, however, provide a structure for the successful adoption of your selected management and development practices.

The reason for the success of the use-case approach is not just that it is a very practical technique to capture requirements from a usage perspective or to design practical user experiences, but it impacts the whole development life cycle. The key use cases—or to be more precise, the key use-case slices (a slice being a carefully selected part of a use case)—assist systematically in finding the application *architecture*. They drive the identification of components or other software elements in software design. They are the elements that must go through testing—and truly support test-driven design. They are the elements to put in the backlog when planning *sprints* or to put on the canvas using Kanban. The use cases of a business are the processes of the business; thus, the advantage of doing business modeling with use cases is that it leads directly to finding the use cases of the system to be developed to support the business. Moreover, use cases help in finding commonalities, which directs the architecture work to achieve software reuse.

There are many more similar values in applying use cases, but most important is the idea of use cases is intuitively graspable. This is a lightweight, lean and agile, scalable, versatile, and easy-to-use approach. Many people who hear about use cases for the first time take them to heart; many start using the term in everyday-life situations without thinking about all the details

that help with so many aspects of software development—the aspects that are the spokes of the software-development wheel in which use cases are the hub (see Figure 1).

Thus, it seems clear use cases have stood the test of time and have a very healthy future.

Principles for Use-Case Adoption

There are six basic principles at the heart of any successful application of use cases. Here, we discuss each one.

Principle 1: Keep it simple by telling stories. Storytelling is how cultures survive and progress; it is the simplest and most effective way to pass knowledge from one person to another. It is the best way to communicate what a system should do and to get everybody working on the system to focus on the same goals.

Use cases capture the goals of the system. To understand a use case, we tell stories. The stories cover how to achieve the goal and how to handle problems that occur on the way. Use cases provide a way to identify and capture all the different but related stories in a simple but comprehensive way. This enables the system's requirements to be easily captured, shared, and understood.

Principle 2: Understand the big picture. Whether the system you are developing is large or small, whether it is a software system, a hardware system, or a business system, understanding the big picture is essential. Without an understanding of the system as a whole, you will find it impossible to make the correct decisions about what to include in the system, what to leave out, what it will cost, and what benefit it will provide.

A use-case diagram is a simple way of presenting an overview of a system's requirements. Figure 2 is the use-case diagram for a simple telephone system. This picture shows all the ways the system can be used, who starts the interaction, and any other parties involved. For example, a calling subscriber can place a local call or a long-distance call to any of the system's callable subscribers. You can also see the users don't have to be people but can be other systems and, in some cases, both (for example, the role of the called subscriber



Use cases can and should be used to drive software development ... They provide a structure for the successful adoption of your selected management and development practices.



might be an answering machine and not a person).

Principle 3: Focus on value. When trying to understand how a system will be used, it is always important to focus on the value it will provide to its users and other stakeholders. Value is generated only if the system is actually used, so it is much better to focus on how the system will be used than on long lists of the functions or features it will offer.

Use cases provide this focus by concentrating on how the system will be used to achieve a specific goal for a particular user. They encompass many ways of using the system: those that successfully achieve the goals and those that handle any problems that may occur.

Figure 3 shows a use-case narrative structured in this way for the cash-withdrawal use case of a cash machine. The simplest way of achieving the goal is described by the basic flow. The others are presented as alternative flows. In this way you create a set of flows that structure and describe the stories, helping to find the test cases that complete their definition.

This kind of bulleted outline may be enough to capture the stories and drive the development, or it may need to be elaborated on as the team explores the details of what the system needs to do.

Principle 4: Build the system in slices. Most systems require a lot of work before they are usable and ready for operational use. They have many requirements, most of which are dependent on other requirements being implemented before they can be fulfilled and value delivered. It is always a mistake to try to build such a system in one go. The system should be built in slices, each of which has clear value to the users.

The recipe is quite simple. First, identify the most useful thing that the system has to do and focus on that. Then take that one thing, and slice it into thinner slices. Decide on the test cases that represent acceptance of those slices. Choose the most central slice that travels through the entire concept from end to end, or as close to that as possible. Estimate it as a team and start building it.

This is the approach taken by Use-Case 2.0, where the use cases are sliced up to provide suitably sized work items, and where the system itself evolves slice by slice.

Although use cases have traditionally been used to help understand and capture requirements, they have always been about more than this. The use-case slices slice through more than just the requirements; they also slice through all the other aspects of the system and its documentation, including the design, implementation, test cases, and test results.

Principle 5: Deliver the system in increments. Most software systems evolve through many generations. They are not produced in one go; they are constructed as a series of releases, each building on the one before. Even the releases themselves are often not produced in one go but evolve through a series of increments. Each increment provides a demonstrable or usable version of the system. This is the way all systems should be produced.

Figure 4 shows the incremental development of a system release. The first increment contains only a single slice—the first slice from use-case 1. The second increment adds another slice from use-case 1 and the first slice from use-case 2. Further slices are then added to create the third and fourth increments. The fourth increment is considered complete and useful enough to be released.

Principle 6: Adapt to meet the team's needs. Unfortunately, there is no one-size-fits-all solution to the challenges of software development; different teams and different situations require different styles and different levels of detail. Regardless of which practices and techniques you select, you need to ensure they are adaptable enough to meet the ongoing needs of the team.

Use-Case 2.0 is designed with this in mind and can be as light as desired. Small, collaborative teams can have very lightweight use-case narratives that capture the bare essentials of the stories. These can be handwritten on simple index cards. Large distributed teams can have more detailed use-case narratives presented as documents. It is up to the team to decide whether or not they need to go beyond the essentials, adding detail in a natural fashion as they encounter problems the bare essentials cannot cope with.

The Use-Case 2.0 Practice

The Use-Case 2.0 practice describes the key concepts to work with, the work



Use cases cover many related stories of varying importance and priority. There are often too many stories to deliver in a single release and generally too many to work on in a single increment.



products used to describe them, and a set of activities.

Concepts to work with. Use-Case 2.0 encompasses the requirements, the system to be developed to meet the requirements, and the tests used to demonstrate the system meets the requirements. At the heart of Use-Case 2.0 are the *use case*, the *use-case slice*, and the *story*.

Use cases capture the requirements, and each use case is scope managed by slicing it up into a set of use-case slices that can be worked on independently. Telling stories bridges the gaps between the stakeholders, the use cases, and the use-case slices. This is how the stakeholders communicate their requirements and explore the use cases. Understanding the stories is also the mechanism for finding the right use-case slices to drive the implementation of the system.

A *use case* is a sequence of actions a system performs that yields an observable result of value to a particular user.

- ▶ That specific behavior of a system, which participates in collaboration with a user to deliver something of value for that user.
- ▶ The smallest unit of activity that provides a meaningful result to the user.
- ▶ The context for a set of related requirements.

Taken together, the set of all use cases gives us all the functional requirements of the system.

The way to understand a use case is to tell stories. These stories cover both how to achieve a goal and how to handle any problems that occur on the way. They help developers understand the use case and implement it slice by slice.

A use case undergoes several defined state changes, beginning with just having its *goal established*, through *story structure understood*, *simplest story fulfilled*, *sufficient stories fulfilled*, to *all stories fulfilled*. The states constitute important waypoints in the understanding and implementation of the use case.

Use cases cover many related stories of varying importance and priority. There are often too many stories to deliver in a single release and generally too many to work on in a single increment. Hence, there is a need for dividing use cases into smaller pieces.

A use-case slice is one or more stories selected from a use case to form a work item that is of clear value to the customer. It acts as a placeholder for all the work required to complete the implementation of the selected stories. The use-case slice evolves through design, implementation, and test.

The *use-case slice* is the most important element of Use-Case 2.0, as it is used not only to help with the requirements, but also to drive the development of a system to fulfill them.

A use-case slice undergoes several state changes, from its initial identification where it is *scoped*, through being *prepared*, *analyzed*, *implemented*, and, finally, *verified*. These states allow for planning and tracking the understanding, implementation, and testing of the use-case slice.

To the casual observer glancing at the states, this might look like a waterfall process. There is a big difference, though, as this involves an individual use-case slice. Across the set of slices all the activities could be going on in parallel. While one use-case slice is being verified, another use-case slice is being implemented, a third is being prepared, and a fourth is being analyzed.

Telling *stories* is how developers explore use cases with stakeholders. Each story of value to the users and other stakeholders is a thread through one of the use cases. The stories can be functional or non-functional in nature.

Part of the use-case narrative, one or more flows and special requirements, and one or more test cases describe a story. The key to finding effective stories is to understand the structure of the use-case narrative. The network of flows can be thought of as a map that summarizes all the stories needed to describe the use case. In the previous cash-machine example in Figure 3, you could identify specific stories such as “Withdraw a standard amount of \$100,” “Withdraw a nonstandard amount of \$75 and get a receipt,” or “Respond to an invalid card.”

Each story traverses one or more flows beginning with the use case at the start of the basic flow and terminating with the use case at the end of the basic flow. This ensures all the stories are related to the achievement of the

same goal, are complete and meaningful, and are complementary, as they all build upon the simple story described by the basic flow.

Work products. Use cases and use-case slices are supported by a number of work products that the team uses to help share, understand, and document them.

A *use-case model* visualizes the requirements as a set of use cases, providing an overall big picture of the system to be built. The model defines the use cases and provides the context for the elaboration of individual use cases.

Use cases are explored by telling stories. Each use case is described by (1) a *use-case narrative* that outlines its stories as a set of flows; and (2) a set of test cases that complete the stories. These can be complemented with a set of special requirements that apply to the whole use case and are often non-functional. These will influence the stories, help assign the right stories to the use-case slices for implementation, and, most importantly, define the right test cases. Supporting information complements the use-case model. This cap-

Figure 4. Use cases, use-case slices, increments, and releases.

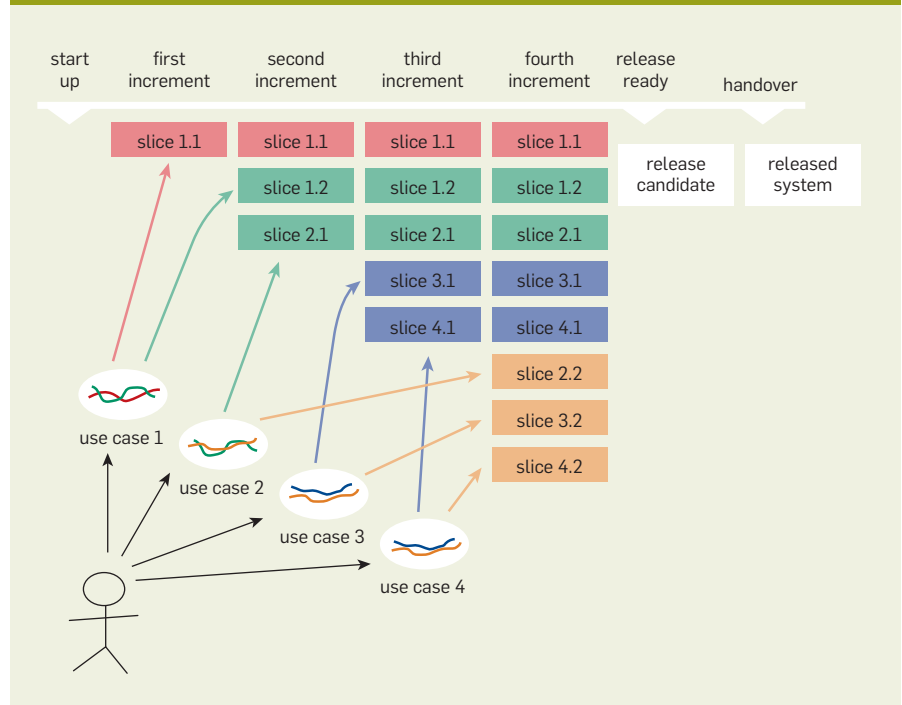


Figure 5. Capturing the properties of a use case and its slices using sticky notes.



tures the definitions of the terms used in the use-case model and when outlining the stories in the use-case narratives. It also captures any systemwide requirements that apply to all of the use cases.

A *use-case realization* can be created to show how the system's elements collaborate to perform a use case. Think of the use-case realization as providing the "how" to complement the use-case narrative's "what." Common ways of expressing use-case realizations include simple tables, storyboards, or sequence diagrams.

Working with the use cases and use-case slices. In addition to creating and tracking the work products, developers need to track the states and properties of use cases and use-case slices. This can be done in many ways and with many tools. The states can be tracked very simply using sticky notes

or spreadsheets. If more formality is required, then one of the many commercially available requirements-management, change-management, or defect-tracking tools can be used.

Figure 5 shows a use case and some of its slices captured on a set of sticky notes.

The use case shown is "7. Browse and Shop" from an online shopping application. Slices 1 and 2 of the use case are based on individual stories derived from the basic flow: "Select and Buy 1 Product" and "Select and Buy 100 Products." Slice 3 is based on multiple stories covering the availability of the various support systems involved in the use case.

The essential properties for a use case are its name, state, and priority. In this case the popular MoSCoW (Must, Should, Could, Would) prioritization scheme has been used. The use cases should also be estimated. Here a sim-

ple scheme of assessing relative size and complexity has been used.

The essential properties for a use-case slice are: a list of its stories; references to the use case and the flows that define the stories; references to the tests and test cases that will be used to verify its completion; and an estimate of the work needed to implement and test the slice. In this example the stories are used to name the slice, and the references to the use case are implicit in the slices number and list of flows. The estimates have been added later after consultation with the team. These are the large numbers toward the bottom right of each sticky note. In this case the team has played Planning Poker to create relative estimates using story points.

The use cases and the use-case slices should also be ordered so that the most important ones are addressed first.

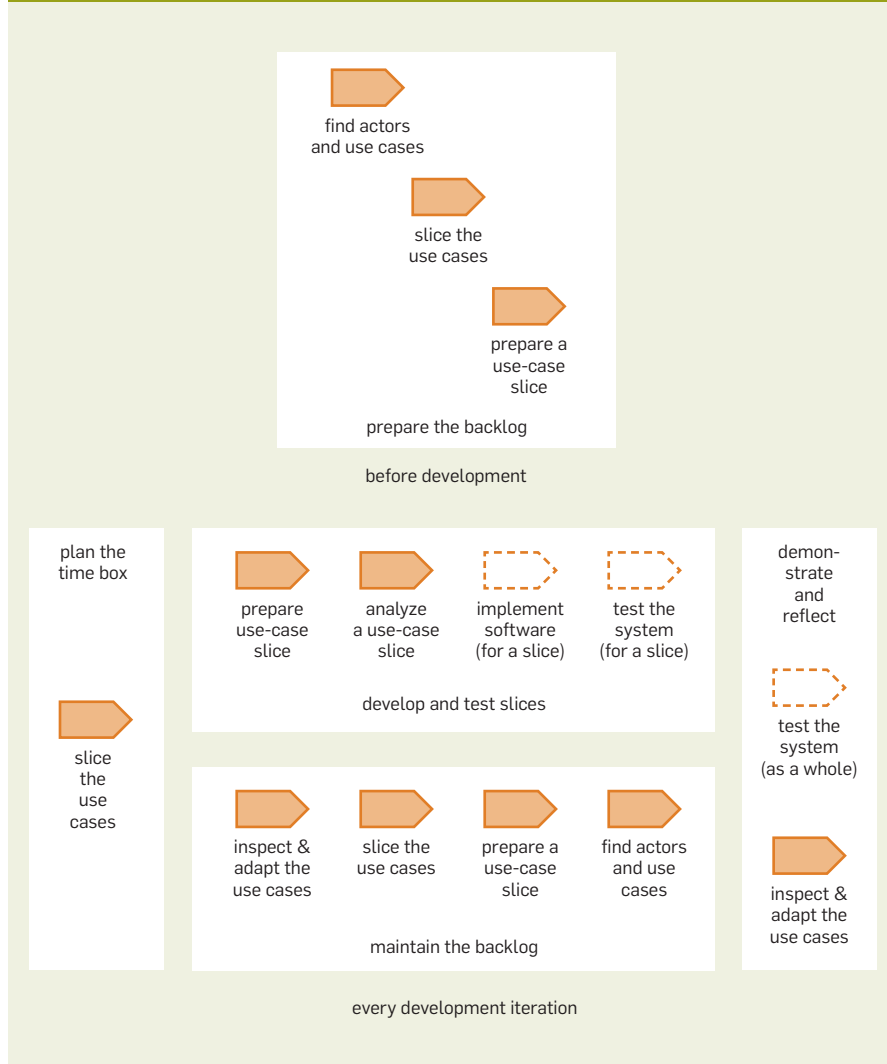
Keeping work products as lightweight as appropriate. All of the work products are defined with a number of levels of detail. The first level defines the bare essentials, the minimal amount of information required for the practice to work. Further levels of detail are defined to help the team cope with any special circumstances they might encounter. This allows small, collaborative teams to have very lightweight use-case narratives defined on simple index cards and large distributed teams to have more detailed use-case narratives presented as documents. The teams can then grow the narratives as needed to help with communication or thoroughly define the important or safety-critical requirements.

The good news is you always start in the same way, with the bare essentials. The team can then continually adapt the level of detail in their use-case narratives to meet their emerging needs.

Things to do. Use-Case 2.0 breaks the work up into a number of essential activities that need to be done if the use cases are to provide real value to the team.

The *Find Actors and Use Cases* activity produces a use-case model that identifies the use cases, which will be subsequently *sliced*. These use-case slices will then be *prepared* by describing the related stories in the use-case narrative and defining the test cases. The slice is *analyzed* to

Figure 6. Use-case activities for iterative development approaches.



work out how the system elements will interact to perform the use case, then *implemented* and *tested* as a slice. Use-Case 2.0 can be considered a form of test-driven development, as it creates the test cases for each slice upfront. Finally, the whole system is *tested* to ensure all the slices work together when combined.

The activities themselves will all be performed many times in the course of your work. Even a simple activity such as *Find Actors and Use Cases* may need to be performed many times to find all the use cases and may be conducted in parallel with, or after, the other activities. For example, while continuing to *Find Actors and Use Cases*, you may also be implementing some of the slices from those use cases found earlier.

As the project progresses, priorities change, lessons are learned, and changes are requested. These can all have an impact on the use cases and use-case slices that have already been implemented, as well as those still waiting to progress. This means there will be an ongoing *Inspect and Adapt* activity for the use cases. This will also adapt the way of working with the Use-Case 2.0 practice to adjust the size of slices or the level of details in work products to meet the varying demands of the project and team.

Using Use-Case 2.0

Many people think use cases are applicable only to user-intensive systems that have a lot of interaction between the human users and the system. This is strange because the original idea for use cases came from telecom switching systems, which have both human users (subscribers, operators) and machine users, in the form of other interconnected systems. Use cases are applicable to all systems that are used—and that means *all* systems.

It is not just for user-intensive applications. In fact, use cases are just as useful for embedded systems with little or no human interaction as they are for user-intensive ones. People are using use cases in the development of all kinds of embedded software in domains as diverse as motor, consumer electronics, military, aerospace, and medical industries. Even real-time process-control systems used for chemical



Use cases are applicable to all systems that are used—and that means *all* systems.



plants can be described by use cases where each use case focuses on a specific part of the plant's process behavior and automation needs.

It is not just for software development. The application of use cases is not limited to software development. They can also help understand business requirements, analyze existing business, design new and better business processes, and exploit the power of IT to transform business. By using use cases recursively to model the business and its interactions with the outside world and model the systems needed to support and improve the business, developers can seamlessly identify where the systems will impact the business and which systems are needed to support the business.

Handling all types of requirements. Although they are one of the most popular techniques for describing systems' functionality, use cases are also used to explore non-functional characteristics. The simplest way of doing this is to capture them as part of the use cases themselves—for example, relating performance requirements to the time taken between specific steps of a use case or listing the expected service levels for a use case as part of the use case itself.

Some non-functional characteristics are subtler than this and apply to many, if not all, of the use cases. This is particularly true when building layered architectures, including infrastructure components such as security, transaction management, messaging services, and data management. The requirements in these areas can still be expressed as use cases—separate use cases focused on the technical usage of the system. These additional use cases are called *infrastructure use cases*,⁸ as the requirements they contain will drive the creation of the infrastructure on which the application will run.

Applicable for all development approaches. Use-Case 2.0 works with all popular software-development approaches, including:

- ▶ Backlog-driven iterative approaches such as Scrum, EssUP, and OpenUP.
- ▶ One-piece flow-based approaches such as Kanban.
- ▶ All-in-one-go approaches such as the traditional waterfall.

Use-case 2.0 and backlog-driven iterations. Before adopting any backlog-driven approach, you must understand what items will go in the backlog. There are various forms of backlogs that teams use to drive their work, including product, release, and project backlogs. Regardless of the terminology used, they all follow the same principles. The backlog itself is an ordered list of everything that might be needed and is the single source of requirements for any changes to be made.

In Use-Case 2.0, the use-case slices are the primary backlog items. The use of use-case slices ensures backlog items are well formed, as they are naturally independent, valuable, and testable. The structuring of the use-case narrative that defines them ensure they are estimable and negotiable, and the use-case slicing mechanism enables them to be sliced as small as needed to support the development team.

When a backlog-driven approach is adopted, it is important to realize the backlog is not built and completed upfront but is continually worked on and refined. The typical sequence of activities for a backlog-driven, iterative approach is shown in Figure 6.

Use-case 2.0 and one-piece flow. One-piece flow is a technique taken from lean manufacturing that avoids the batching of the requirements seen in

the iterative and waterfall approaches. Each requirements item flows quickly through the development process, but to work effectively, this technique needs small, regularly sized items. Use cases would be too irregularly sized and too big to flow through the system. Use-case slices, though, can be sized appropriately and tuned to meet the needs of the team.

One-piece flow does not mean there is only one requirements item being worked on at a time or that there is only one piece of work between one workstation and the next. Enough items need to be in the system to keep the team busy. Work-in-progress limits are used to level the flow and prevent any wasteful backlogs from building up. Figure 7 shows a simple Kanban board for visualizing the flow of use-case slices.

The work-in-progress limits are shown in red. Reading from left to right, you can see that slices have to be identified and scoped before they are input to the team. In this figure the work-in-progress limit is five, and the customers, product owner, or requirements team that are the source of the requirements try to keep five use-case slices ready for implementation at all times.

Note there is no definitive Kanban board or set of work-in-progress limits; it is dependent on team structure and working practices. The board and work-in-progress limits should

be tuned the same as practices. The states for the use-case slices are a great aid to this kind of work design as they can clearly define what state the slice should be in when it is to be handed to the next part of the chain.

Use-case 2.0 and waterfall. For various reasons you may need to develop software within the constraints of some form of waterfall governance model. This typically means some attempt will be made to capture all the requirements upfront before they are handed over to a third party for development.

In a waterfall approach the use cases are not continually worked on and refined to allow the final system to emerge but are defined in one go at the start of the work.

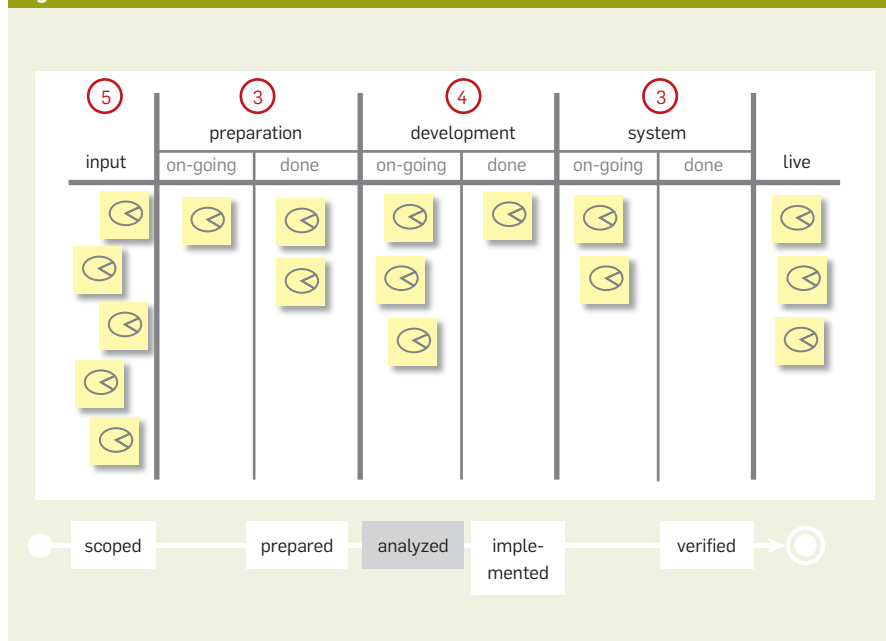
The one-thing-at-a-time nature of the waterfall approach means that the makeup of the team is continually changing over time, so the ability to use face-to-face communication to share the stories is very limited. To cope with this, you need to turn up the level of detail on the work products, going beyond the bare essentials.

Use-case 2.0—It is not just for one type of team. Another important aspect of Use-Case 2.0 is its ability to adapt to existing team structures and job functions while encouraging teams to eliminate waste and increase efficiency. To this end, Use-Case 2.0 does not predefine any particular roles or team structures, but it does define a set of states for each of the central elements (the use case and the use-case slice).

As illustrated by the discussion on Use-Case 2.0 and one-piece flow, the states indicate when the items are at rest and could be handed over from one person or team to another. This allows the practice to be used with teams of all shapes and sizes, from small cross-functional teams with little or no handovers to large networks of specialist teams where each state change is the responsibility of a different specialist. Tracking the states and handovers of these elements allows the flow of work through the team (or teams) to be monitored, and teams to adapt their way of work to improve their performance continuously.

Scaling to meet your needs—in, out, and up. No one predefined approach fits everyone, so the use of Use-Case 2.0

Figure 7. Use-case slices on a Kanban board.



needs to be scaled in a number of different dimensions:

- Use cases scale in to provide more guidance to less-experienced practitioners (developers, analysts, testers, among others) or to practitioners who want or need more guidance.

- They scale out to cover the entire life cycle, covering analysis, design, coding, and test, as well as operational usage and maintenance.

- They scale up to support large and very large systems such as systems of systems: enterprise systems, product lines, and layered systems. Such systems are complex and typically developed by many teams working in parallel at different sites, possibly for different companies, and reusing many legacy systems or packaged solutions.

Regardless of the complexity of the system under development, the development team always starts in the same way by identifying the most important use cases and creating a big picture summarizing what needs to be built. Use-Case 2.0 can then be adapted to meet the emerging needs of the team. In fact, the Use-Case 2.0 practice insists you continuously inspect and adapt its usage to eliminate waste, increase throughput, and keep pace with the ever-changing demands of the team.

User Stories and Use Cases—What Is the Difference?

The best way to answer this question is to look at the common properties of user stories and use cases—the things that make both work well as backlog items and enable both to support popular agile approaches such as Scrum, Kanban, test-driven development, and specification by example.

Use-Case slices and user stories³ share many common characteristics. For example:

- They both define slices of the functionality that teams can get done in a sprint.

- They can both be sliced up if they are too large, resulting in more, smaller items.

- They can both be written on index cards.

- They both result in test cases that represent the acceptance criteria.

- They are both placeholders for a conversation and benefit from the

three Cs invented by Ron Jeffries: card, conversation, and confirmation.

- They can both be estimated with techniques such as Planning Poker.

So, given they share so many things in common, what is it that makes them different? Use cases and use-case slices provide added value:

- A big picture to help people understand the extent of the system and its value.

- Increased understanding of what the system does and how it does it.

- Better organization, understanding, application, and maintenance of test assets.

- Easy test-case generation and analysis.

- Support for ongoing impact analysis.

- Active scope management allowing easy focus on providing the minimal viable product.

- Flexible, scalable documentation to help cope with traceability or other contractual constraints.

- Support for simple systems, complex systems, and systems of systems.

- Easier identification of missing and redundant functionality.

The question remains: Which technique should you use that is very context dependent, once you go beyond personal preferences? Consider the following factors: how much access is there to the SMEs (subject matter experts); and how severe will requirements errors be if they escape to a live environment.

The sweet spot for user stories is achieved when there is easy access to a SME and the severity of errors is low. Use cases and use-case slices are more suitable when there is no easy access to a SME or when error consequences are high. Since the use-case approach can scale down to the sweet spot of user stories, however, you may still want to apply them. If the subject system will always be in the sweet spot of user stories, then user stories are fine, but if you expect it to grow outside that area, you should consider use cases and use-case slices.

Use-Case 2.0 exists as a proven and well-defined practice that is compatible with many other software-development practices such as continuous

integration, intentional architecture, and test-driven development. It also works with all popular management practices. In particular, it has the lightness and flexibility to support teams that work in an agile or lean fashion. It also has the completeness and rigor required to support teams that work in a more formal or waterfall environment.

More details about the fully documented Use-Case 2.0 practice are available at <http://www.ivarjacobson.com>. 

Related articles on queue.acm.org

Agile and SEMAT - Perfect Partners

Ivar Jacobson, Ian Spence, and Pan-Wei Ng
<http://queue.acm.org/detail.cfm?id=2541674>

Case Study: UX Design and Agile: A Natural Fit?

<http://queue.acm.org/detail.cfm?id=1891739>

The Cost of Virtualization

Ulrich Drepper

<http://queue.acm.org/detail.cfm?id=1348591>

References

1. Booch, G., Jacobson, I., Rumbaugh, J. *The Unified Modeling Language Reference Manual*, second edition. Addison-Wesley Professional, 2004.
2. Cockburn, A. *Writing Effective Use Cases*. Addison-Wesley Professional, 2001.
3. Cohn, M. *User Stories Applied*. Addison-Wesley Professional, 2004.
4. Constantine, L. and Lockwood, L. *Software for Use*. Addison-Wesley Professional, 1999.
5. Jacobson, I. Case for aspects, part II. *Software Development Magazine* (Nov. 2003): 42-48.
6. Jacobson, I. Object-oriented software development in an industrial environment. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications Conference* (1987).
7. Jacobson, I., Christerson, M., Johnsson, P. and Overgaards, G. *Object-Oriented Software Engineering: A Use Case-Driven Approach*. Addison-Wesley Professional, 1992.
8. Jacobson, I. and Ng, P.W. *Aspect-oriented Software Development with Use Cases*. Addison-Wesley Professional, 2005.
9. Slama, D., Puhlmann, F., Morrish, J. and Bhatnagar, R. *Enterprise Internet of Things*, 2015; <http://enterprise-Internet of Things.org/book/enterprise-Internet of Things/>.

Ivar Jacobson is a father of components and component architecture, modern business engineering, the Unified Modeling Language, and the Rational Unified Process. His latest contribution is a formal practice concept that promotes practices as the “first-class citizens” of software development. Jacobson is also one of the founders of the SEMAT (Software Engineering Method and Theory) community.

Ian Spence is CTO at Ivar Jacobson International and the team leader for the development of the SEMAT (Software Engineering Method and Theory) kernel. An experienced coach, he has introduced hundreds of projects to iterative and agile practices.

Brian Kerr is an experienced agile coach, consultant, and change agent, and is a principal consultant at Ivar Jacobson International. He works with teams and organizations, helping them adopt key software-development practices in a pragmatic and sustainable way.

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

DOI:10.1145/2830565

Car automation promises to free our hands from the steering wheel but might demand more from our minds.

BY STEPHEN M. CASNER, EDWIN L. HUTCHINS, AND DON NORMAN

The Challenges of Partially Automated Driving

AUTONOMOUS CARS PROMISE to give us back the time we spend in traffic, improve the flow of traffic, reduce accidents, deaths, and injuries, and make personal car travel possible for everyone regardless of their abilities or condition. But despite impressive demonstrations and technical advances, many obstacles remain on the road to fully autonomous cars.²⁰ Overcoming the challenges to enabling autonomous cars to safely operate in highly complex driving situations may take some time.

Manufacturers already produce partially automated cars, and a spirited competition to deliver the most sophisticated ones is under way. Cars that provide high levels of automation in some circumstances (such as

highway driving) have already arrived in the marketplace and promise to be in the hands of a large number of car owners in the next few years.

What does increasing automation require of drivers? The role of the driver in the extreme cases of fully manual

» key insights

- Driving a car is becoming a task shared between humans and technology, but are humans ready to just push a button and let the computers do the driving?
- Human-computer interaction issues abound when car automation systems attempt to give drivers advice or assume control of the vehicle.
- Even as some drivers are attentive behind the wheel and others lured deeper into distraction, all must be ready to take control when automation encounters corner cases.





IMAGE COURTESY OF VOLVO CAR GROUP GLOBAL NEWSROOM

or fully autonomous driving is clear. In manual cars, people drive, and in fully autonomous cars they do not drive. But what is the role of a driver in a partially automated car in which some of the driver's responsibilities are replaced by computers, some of the time? Partial automation makes us part driver and part passenger, having to deal with the familiar problem of working together with computing systems. Even though totally autonomous driving will arrive someday, the transition will be difficult, especially during the period when the automation is both incomplete and imperfect, requiring the human driver to maintain oversight and sometimes intervene and take closer control.²⁸

Here, we review two kinds of emerging car automation systems and

discuss the challenges drivers will likely face when expected to work cooperatively with them behind the wheel. These automation systems range from those that offer informational assistance to drivers to those that can assume control of the vehicle for extended stretches of time—or even seize control of the vehicle when the driver wanders into unsafe situations. We draw on the state of the art in driving research, along with decades of previous work that examined the safety effects of automation as it was gradually introduced in the airline cockpit. We discuss a variety of challenges we expect to arise as automation assumes increasing responsibility for driving tasks once performed solely by humans. Some problems seem counterintuitive and some paradoxical, with

few of them lending themselves to simple solutions. In the end we invite the reader to consider the evidence we present and decide whether drivers are ready to “go on autopilot” behind the wheel of the next generation of cars.

Provide Advice but Leave the Driver in Charge

The first kind of automobile automation to arrive does not seem quite like automation at all. It does not take over the controls of the car. Rather, it leaves the driver to perform the driving task while it gives advice.

Navigation systems. GPS navigation has already automated the routine but labor-intensive (and often distracting) task of figuring out how to get from point A to point B. Driver reliance on paper maps, mental representations,

and out-the-window scanning for navigational clues is being replaced by a combination of moving maps, arrow indicators, verbal instructions, and head-up displays. These devices may seem simple and the advice they provide useful, but a closer examination shows they give rise to numerous problems,⁷ many involving safety.

Inattention. Navigation systems must be programmed, and these interactions pull drivers' attention away from the task of driving. Early research prompted the National Highway Traffic Safety Administration (NHTSA) to issue guidelines stating any interaction with in-car information systems should require a maximum of two seconds at a time, and no more than 12 seconds total.⁴⁰ However, 2013 research by Strayer et al.³⁶ found interacting with voice-controlled navigation systems can be just as distracting as manually operated systems.

Navigation systems can also give rise to a second kind of inattention. When a navigation system performs well over extended periods, drivers may no longer feel they need to pay close attention. Indeed, many psychological studies show people have trouble focusing their attention when there is little or nothing to attend to. In such situations, they tend to reduce their active involvement and simply obey the automation. There is already ample evidence drivers disengage from the navigation task when the automation is programmed to lead the way.²⁶ But how pervasive is this? Casner and Schooler¹⁰ found even well-trained airline pilots report engaging in copious amounts of task-unrelated thought, or "mind wandering," when an advanced navigation system is being used and all is nominally going to plan.

Brittleness. GPS navigation shows us how automation systems can be brittle, solving most problems with ease, until they encounter a difficult, unusual case, and then do not. Consider the case of one driver in England whose navigation system commanded a turn over a cliff when the database on which it relied mistook a footpath for a road.³ Small database errors with large consequences led one early automation human factors expert to coin the phrase "blunders made easy."³⁹

Trust. We might ask what prompted this driver to follow such an instruc-

Warning systems can lead pilots and drivers alike into trouble when they fail to alert and also when they alert too much.

tion. Drivers can presumably tell the difference between a road and a cliff. It is not, alas, that easy. Studies in aviation have shown automation systems earn our trust following periods of impeccable performance, sometimes to the point we feel that the automation knows best.²² Although it is tempting to explain away accidents as isolated examples of incompetent drivers, our experience in aviation again tells us different. Well-trained pilots are prone to the same sorts of mistakes; for instance, in 1995, the crew of a Boeing 757 flew into a mountain near Buga, Colombia, after following the directions given by their erroneously programmed flight-management system.

Quality of feedback. Others have pointed out the amount and quality of feedback provided by systems like GPS units can make all the difference. Some GPS navigation units use their visual displays to show the vehicle is positioned in the center of the road but not where the road leads next or even if it is not a road at all. With limited information about context and surroundings, it is easy for drivers to miss important clues when things go wrong. Some have proposed designing navigation systems to more closely match the way people naturally help each other find their way in a car.¹⁷

Skill atrophy. There is good evidence that cognitive skills erode when not practiced regularly. Though we are aware of no long-term studies of navigation-skill atrophy in drivers, Casner et al.⁹ found significant atrophy in the navigation skills of airline pilots following extended use of a computerized navigation system.

Navigation systems are an excellent example of technology introduced to automate a task for which people already seemed reasonably competent. Yes, drivers got lost before the introduction of navigational systems, but they seldom led to safety-critical incidents. GPS navigation has introduced many human factors complications we did not anticipate.

Driver warning systems. Some kinds of information-automation systems tell us when we are doing (or are about to do) something wrong. Speed-limit alarms can alert us when we inadvertently exceed a pre-set speed limit. Lane-departure warning systems alert

us when we drift from our lane or attempt to change lanes when the target lane is occupied by another vehicle. But such advisory systems are not without their limitations.

Complacency. One unintended consequence of alerts and alarm systems is some drivers may substitute the secondary task of listening for alerts and alarms for the primary task of paying attention. Wiener³⁸ termed this effect “primary-secondary task inversion,” pointing out the problem is commonplace among experienced airline pilots. Palmer et al.²⁹ described many cases in which pilots missed an assigned altitude when an altitude alert (designed to advise pilots of an upcoming altitude) failed to sound. It is easy to imagine drivers allowing themselves to be distracted for prolonged periods and relying on alert systems to call when trouble pops up.

Nuisance alerts. Warning systems can lead pilots and drivers alike into trouble when they fail to alert and also when they alert too much. In aviation, alerts and alarms given in situations pilots do not find alarming cause them to ignore the alerts.⁵ It is easy to imagine our own reactions to a system that continuously reminds us we are driving five miles per hour over the speed limit. A second problem with alerts is they can be startling. Although human factors engineers have learned to minimize the startle effect of unexpected sound by adjusting loudness, rise time, and other characteristics, the physiological responses to unanticipated signals are difficult to avoid. Lastly, when multiple alerts sound simultaneously, the resulting cacophony can overload and confuse.¹³ Solutions range from trying to use different modalities for different alerts to trying to prioritize the various alerts, aiming to present only the most significant. An alternative approach would be to present a single holistic display—whether visual, auditory, or haptic or all three—that would present a single cohesive conceptual model of the situation. All these ideas are still at the research stage.

Short timeframes. The automobile is far more dangerous than an airplane in several respects. One is the rapidity with which a response is required. In an airplane flying at cruising altitude of 10 km–12 km, the pilots might have min-

utes in which to respond. In a car, the available time can sometimes amount to a fraction of a second. Drivers must construct an understanding of the situation, decide how to respond, and do it successfully in short order. Laboratory studies of driver reactions to rear-end-collision alerts show the effectiveness of these alerts falls off quickly when alert times are short.^{25,44}

Summary. Although it sounds simple enough, the idea of drivers “being informed” by automated systems is not straightforward. On the one hand, systems must keep drivers informed of the driving conditions, including the state of the automobile, the road, and other cars. On the other, too much information can lead to distraction and a failure to attend to any of it.

Assume Control of the Vehicle

A second kind of automation can directly control all or part of an automobile. The arrival of such automation represents a steady progression from the totally manual cars we have today to fully automated cars tomorrow. To provide a regulatory framework for the development and deployment of automation that can operate a car’s controls, NHTSA formalized levels to describe the degree of automation. Level 0 is a totally manual car. Level 4 is a fully self-driving car that requires nothing from its occupants, or even that any occupants be present. We describe these levels, along with the human factors complications known to be associated with increasing automation of this type.

Level 0 (the manual car). The Level 0 car is entirely manual. Why discuss cars with no automation? Because the improved stability of modern cars and the smoothness of paved roads already allow drivers to take their eyes off the road and their hands off the steering wheel, giving us a preview of a first problem with vehicle-control automation.

Inattention. Level 0 cars already reduce driving to a remarkably mundane task, sometimes requiring little attention from the driver and luring the driver into distraction.

The original behind-the-wheel diversion was talking with passengers. Some studies conclude in-car conversations can interfere with driving.^{4,14} Yet other studies demonstrate a “two heads are

better than one” effect, increasing the total amount of vigilance in a car when conversation is carefully managed.^{32,37} These studies reiterate that drivers and passengers have a shared understanding of the driving context and may be able to modulate their talking as the situation demands.¹⁸

Entertainment systems are a known distraction. Aside from the driver attention required to tune radio stations and select music, studies demonstrate listening to music takes a toll on driving performance.⁶

Personal electronics devices (such as mobile phones) provide even more distraction. Why do drivers keep talking, texting, emailing, posting, and even video calling?²⁴ Roy and Liersch³¹ showed people who engage in such behaviors believe they have superior multitasking skills. Unfortunately, the evidence does not support this view. Multitasking is done by rapidly switching between tasks, not only taking attention from driving but also adding a heavy mental load in reestablishing the context of each task as it is reengaged. Interactions with devices (such as smartphones) can lure people into long excursions away from the driving task. The results can be tragic. In a 2009 smartphone-related fatality, a driver drove 84 miles per hour into the rear of a stopped car, with no brakes applied before impact.⁴¹ Revisiting the effect of having passengers in the car, a 2009 study provides evidence passengers can help limit a driver’s use of a personal electronics device behind the wheel.¹²

Even without the distraction of our technologies, drivers’ minds inevitably wander. He et al.²¹ and Yanko and Spalek⁴² showed the prevalence of mind wandering behind the wheel of a conventional car and its effect on driving performance. Knowing where you are going only seems to make the problem worse.⁴³

Level 1 (function-specific automation). NHTSA’s Level 1 refers to cars that use automation to operate a single control. Many modern cars incorporate automated safety systems (such as anti-lock braking, brake assist, electronic stability control, and electronic traction control), but these systems operate only when needed and that operation is largely invisible to the driver.

For example, electronic traction control brakes the wheels individually and transfers more torque to wheels with traction; this allows a driver to pull away and accelerate on slippery surfaces. Braking drive wheels individually is not something a driver could do. These automated systems operate in the control loop together with the driver, augmenting the control functions.

A cruise-control system that maintains a target speed is another example of function-specific automation. It is technologically simpler than driving safety systems but from a human factors perspective is much more complex. It requires explicit activation and deactivation by the driver and on the open road frees the driver from having to attend to vehicle speed. Cruise control automates the feedback loop that controls speed. This creates the possibility of the driver mentally drifting out of the feedback loop.³⁵

The tiring task of doing a little less. Dufour¹⁵ showed relieving drivers of even one aspect of the driving task results in reports of increased driver drowsiness and reduced vigilance when driving on open stretches of road. But the effects do not stop there. Dufour also showed drivers take more time to respond to sudden events when they use cruise control. The message is clear. If you take drivers out of the role of active control, it is difficult to get them back in when they are needed.

One solution to the problems introduced by today's cruise-control systems is to add more automation. Aviation human factors expert Earl Wiener termed this the "one more computer" solution. What will be the effect of adding yet another computer to address the problems of driver inattention? Our experience with automation in other domains tells us that rising levels of automation will lead to declining levels of awareness.^{9,16} Unfortunately, adding computers to the mix is precisely what is being done. This brings us to Level 2 automation.

Level 2 (combined function automation). NHTSA's Level 2 refers to cars that use automation to control two or more functions of the driving task at once. A key feature of Level 2 automation is it is generally capable of fully controlling the vehicle for limited periods in restricted situations

(such as following another car during uneventful freeway cruising or during traffic jams). Most automated systems at Level 2 and above assume control loops that operate without driver involvement. Two examples of Level 2 automation are the highway pilot and the traffic jam pilot systems being marketed today. They combine an adaptive cruise control system capable of adjusting the target cruise speed when a car ahead slows down or speeds up, along with an automatic lane-keeping system that maintains the vehicle within a chosen lane. The initial release of these systems gave drivers hands-free following and lane keeping for up to 10 to 15 seconds, but today's systems can keep a car driving without attention for tens of minutes.

NHTSA Level 2 assumes the human driver will continue to closely monitor the automation as it follows the car ahead. Manufacturers differ in their requirements for drivers to keep their hands on the steering wheel. Some simply require driver hands to be near the steering wheel in the case a driver takeover is required on short notice.

Inattention (again). Level 2 automation could invite drivers to take their attention away from the driving task for longer stretches of time. If we consider the temptation of handheld devices that are already in use in manual cars today, it is not difficult to imagine where this might lead. As automation becomes more able and reliable, drivers will inevitably do things other than pay attention to driving. They may let their minds wander or even read or take a nap. Distracted drivers today periodically glance up from their handheld devices. Will they continue to glance up with the same frequency as cars provide more sophisticated automation? Driving researchers are studying these situations, and the results are not encouraging.⁸

More about feedback. The problem of reengaging drivers to assume active control of the vehicle is quite complex with Level 2 automation. The driver must be able to determine, at any moment, what driving functions are being handled by the automation and what functions remain the responsibility of the driver. Eye-tracking studies of airline pilots reveal they persistently misremember the state of the automa-

tion, even when they themselves set up the state. They rely on their memory of having pushed a button and habitually ignore system-status displays that tell the real story.³⁴ Incidents in which pilots pressed a button to engage a speed-control function only to later see their speed increase or decrease unexpectedly are commonplace. Pilots sometimes erroneously assume automation functions are available for use when they are not. Automation functions sometimes quietly turn themselves off for no apparent reason. Though flight instructors try to teach pilots to assess the system state by looking at the big picture, problems still abound.

But Level 2 automation can be used in another way—offer assistance during manual driving when the driver wanders into dangerous situations. We have discussed the limitations of warning systems that provide hints to the driver about what to do next. Rather than giving advice, Level 2 automation can be used to simply take control of a vehicle in dire situations. Which is better: give advice or simply take control? Itoh and Inagaki²³ compared advice-giving and takeover approaches during inadvertent lane departures and found the takeover approach resulted in greater overall safety. NHTSA²⁷ estimated electronic stability control systems saved 1,144 lives in the U.S. in 2012 alone. These systems monitor the position of the steering wheel and the actual direction of travel of the vehicle. When the system senses a discrepancy between the two—or loss of steering control—it automatically applies differential braking to all four wheels to counter skidding conditions.

Perhaps the most compelling argument in favor of driver takeover systems comes up when we acknowledge more than half of all fatal accidents in 2009 in the U.S. happened in the presence of aggressive or angry driving.¹ Imagine the life-saving potential of a system that blocks a driver's reckless attempt to step on the gas, use the shoulder of the road, or come dangerously close to another vehicle in an attempt to pass it.

Though these examples make a solid case for "automation knows best," we have also seen many examples in aviation in which pilots fought automation for control of an aircraft. In 1988, during an air show in Habsheim, France,

in which an Airbus A320 aircraft was being demonstrated, the automation placed itself in landing configuration when the crew did a flyby of the crowd. Knowing there was no runway there, the flight crew attempted to climb. Automation and flight crew fought for control, and the autoflight system eventually flew the airplane into the trees. In this case, the flight crew knew best but its inputs were overridden by an automated system. Now imagine a case in which the GPS suggested a turn into a road that has just experienced a major catastrophe, perhaps with a large, deep hole in what would ordinarily be a perfectly flat roadway. Now imagine an automated car that forces the driver to follow the instruction. In such cases we want the person or thing that is indeed right to win the argument, but as human and machine are both sometimes fallible, these conflicts are not always easy to resolve.

How do we address the problems associated with Level 2 automation? One solution is to largely eliminate the need for attention and understanding from drivers by adding even more automation, bringing us to Level 3 automation.

Level 3 (limited self-driving automation). NHTSA's Level 3 refers to cars that use automation to control all aspects of the driving task for extended periods. Level 3 automation does not require the driver's constant attention, only that the automation provides drivers a comfortable transition time when human intervention is needed. When drivers are needed, the system relies on what is called "conditional driver take-over" in which drivers are summoned and asked to intervene.

Rapid onboarding. One challenge for designers is that people have great difficulty reestablishing the driving context, or as psychologists call it "rapid onboarding." To make matters worse, automation often fails when it encounters unexpected problems, leaving the driver with only a short time to respond. Driving researchers have begun to show drivers' onboarding times grow quickly when high levels of automation are combined with complex situations.¹⁹ Worse, studies of airline pilots responding to such unexpected events inspire little confidence.¹¹

Manual skill atrophy. Prolonged use of automation leads to deterioration of



Tomorrow, we will have accidents that result when drivers are caught even more unaware.



skills. Airline pilots who use high levels of automation in an airline cockpit continually complain about it. Casner et al.⁹ found cognitive skills (such as navigating and troubleshooting) were quick to deteriorate in the absence of practice. Fortunately, they also found "hands on" skills are remarkably resistant to forgetting. Although this sounds encouraging, cognitive skills are needed first to determine what manual operations are required.

Increasing complexity. Automation systems grow to be quite complex and, as a result, difficult to understand, especially by untrained drivers. Even in aviation, where pilots are well trained, automation systems are complex enough to leave them, not with definitive knowledge about how the systems work, but rather working theories that evolve over time. Pilots and researchers alike talk of "automation surprises" in which the automation does something unexpected, leaving the flight crew having to sort it out.³³ The National Transportation Safety Board ruled a contributing factor in the July 6, 2013 Asiana Airlines Flight 214 crash at San Francisco International Airport was a complex user interface to the airplane's autoflight system that was insufficiently understood and perhaps overly trusted by the flight crew.

The complexity issue is likely to grow. Modern sensor technology makes it possible for vehicles to communicate with each other and negotiate joint maneuvers involving several vehicles (such as multi-vehicle collision avoidance). Drivers will be unable to monitor these communications, in part because they occur frequently, at high speed. Almost anything a driver does in such situations is likely to degrade the automatically computed solution. This is fertile ground for what Perrow³⁰ called "systems-level" or "normal" accidents, where accidents are not caused by the actions of an individual but emerge from the behavior of an entire system.

One of the most daunting challenges will happen when we reach the crossover point where automation systems are not yet robust and reliable enough to operate without humans standing by to take over but yet are too complex for people to comprehend and intervene in a meaningful way.

Automation that operates the controls of a vehicle could magnify the problem of maintaining driver attention, along with the consequences of lapses in driver attention. When drivers are unexpectedly asked to reassume control of the car, they are likely to struggle to get back “in the loop” to assess the situation and be able to respond in time. Some of these struggles arise from having to gather the details of the vehicle’s situation, while others arise from the complexity of the automation itself—when the details of how the automation works might elude the driver’s understanding.

Level 4 (full automation). At Level 4, the car is completely automatic. Once Level 4 has been achieved and fully accepted by the driving public, we expect cars will simply become transportation pods, without any manual controls at all except as a means of instructing the vehicle about the desired destination and giving instructions about the drive itself, much as one instructs a chauffeur-driven car today. There will be no need for steering wheel or brake, though there might always be an emergency stop button. Fully automated cars will be just that. There will be no role for drivers, and no need for driving tests, age limits, and concern about sobriety or distraction.

Conclusion

A steady march toward the automated car is clearly under way. The NHTSA levels reflect the belief that automated systems will progressively assume more and more driving tasks until all are done by automation and none are left in the hands of drivers. But due to the many remaining obstacles, and the rate at which cars are replaced on the roadways worldwide, the transition to fully automated driving for a majority of the public will take decades. The safety challenges of partially automated driving will be significant and, at least as of today, underestimated. We thus draw two sets of conclusions, one for drivers, one for car designers.

Drivers. Because car automation systems will gradually increase in capability, drivers will still be required to pay full attention to the driving situation, even if they are not required to actually do anything. They may be required to take control under unan-



To help maintain driving skill, wakefulness, or attentiveness, car interfaces might periodically ask the driver to assume manual control.



nounced and unexpected circumstances, usually with little time to react. Our experience in aviation tells us this transition will not go smoothly for a cadre of cursorily trained drivers in an environment in which milliseconds might mean the difference between life and death. Drivers will expect their cars’ automation systems to function as advertised, and the systems will do so most of the time. And with automation in charge, drivers will learn they can attend more and more to non-driving activities. They will grow to trust the automation to take care of them while they do other things. They will count on automated warnings to alert them when their attention is needed. When the unexpected happens and driver attention is needed with little or no warning, a new kind of accident may emerge, in significant numbers. Today, we have accidents that result when drivers are caught unaware. Tomorrow, we will have accidents that result when drivers are caught even more unaware. We can only echo a plea that is being made to drivers today: Set personal electronic devices aside, resist any temptation to become distracted, and remain focused on the road.

We should also look to the accident record we have today and wonder if, despite such problems, partial automation may not make a corresponding reduction in existing types of accidents. We could see dramatic safety enhancements from automated systems that share the control loop with the driver (such as brake-assist systems and lane-keeping assistance) and especially from systems that take control from the hands of aggressive, distracted, or intoxicated drivers. It is entirely possible that reductions in these categories of accidents could match or even outnumber any increase in accidents caused by other unexpected problems with automation. We expect the most serious problems to arise in systems that take the driver out of the loop, yet these are the very systems drivers want, precisely because they free the driver to do something other than drive.

Car designers. We learned in aviation that interface design indeed has a significant influence on the safety outcomes of automated systems. Drivers will need controls and displays that

address the many problems we have outlined here. Driver interfaces will need to simplify and make transparent the process of passing control of the vehicle between driver and automation. The interface must further make clear the process of determining who or what is controlling the car and what the automation is doing and what it plans to do next. A particularly difficult interface challenge presents itself when a driver attempts to engage an automation function that is not ready to be engaged. We have seen too many cases in which experienced well-trained pilots pressed a button and assumed all would go according to plan, only to be surprised later. To help maintain driving skill, wakefulness, or attentiveness, car interfaces might periodically ask the driver to assume manual control.

Given the great time and expense required to design and certify a new airplane, and the often-30-year periods between airline-equipment refreshes, the aviation industry remains limited in its ability to iteratively develop and test new interface concepts. The car industry may have the luxury of being more exploratory in its design efforts and consider many more possible ways of combining human driver and car automation.

Automation in the car is here. In the coming decades, we will all participate in driving research as an enormous uncontrolled experiment takes place on our streets and highways. But with proper care and design, we can help minimize accidents caused by the presence of automation, from too much trust, lack of attention, and atrophied skills. Lianne Bainbridge of University College London pointed out in her classic 1983 paper “Ironies of Automation,”² “... the more advanced a control system is, so the more crucial may be the contribution of the human operator.”

References

1. AAA. *Aggressive Driving: Research Update*. American Automobile Association Foundation for Traffic Safety, Washington, D.C., 2009; <https://www.aaafoundation.org/sites/default/files/AggressiveDrivingResearchUpdate2009.pdf>
2. Bainbridge, L. Ironies of automation. *Automatica* 19, 6 (1983), 775–779.
3. BBC News. Man follows sat nav to cliff edge. *BBC News* (Mar. 25, 2009); http://news.bbc.co.uk/1/hi/fr-12/hi/uk_news/england/bradford/7962212.stm
4. Bergen, B., Medeiros-Ward, N., Wheeler, K., Drews, F., and Strayer, D. The cross-talk hypothesis: Why language interferes with driving. *Journal of Experimental Psychology: General* 142, 1 (2013), 119–130.
5. Breznitz, S. *Cry Wolf: The Psychology of False Alarms*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.
6. Brodsky, W. and Stor, Z. Background music as a risk factor for distraction among young novice drivers. *Accident Analysis & Prevention* 59 (Oct. 2013), 382–393.
7. Brown, B. and Laurie, E. The normal natural troubles of driving with GPS. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Austin, TX, May 5–10). ACM Press, New York, 2012, 1621–1630.
8. Carsten, O., Lai, F.C.H., Barnard, Y., Jamson, A.H., and Merat, N. Control task substitution in semi-automated driving: Does it matter what aspects are automated? *Human Factors* 54, 5 (2012), 747–761.
9. Casner, S.M., Geven, R.W., Recker, M.P., and Schooler, J.W. The retention of manual flying skills in the automated cockpit. *Human Factors* 56, 8 (2014), 1506–1516.
10. Casner, S.M. and Schooler, J.W. Thoughts in flight: Automation use and pilots’ task-related and task-unrelated thought. *Human Factors* 56, 3 (2014), 433–442.
11. Casner, S.M., Geven, R.W., and Williams, K.T. The effectiveness of airline pilot training for abnormal events. *Human Factors* 53, 3 (2013), 477–485.
12. Charlton, S.G. Driving while conversing: Cell phones that distract and passengers who react. *Accident Analysis & Prevention* 4 (Feb. 2009), 160–173.
13. Cummings, M.L., Kilgore, R.M., Wang, E., Tijerina, L., and Kochhar, D.S. Effects of single versus multiple warnings on driver performance. *Human Factors* 49, 6 (2007), 1097–1106.
14. Drews, F.A., Pasupathi, M., and Strayer, D.L. Passenger and cell phone conversations in simulated driving. *Journal of Experimental Psychology: Applied* 14, 4 (2008), 392–400.
15. Dufour, A. Driving assistance technologies and vigilance: Impact of speed limiters and cruise control on drivers’ vigilance. Presentation at the International Transport Forum (Paris, France, Apr. 15, 2014).
16. Endsley, M.R. and Kiris, E.O. The out-of-the-loop performance problem and level of control in automation. *Human Factors* 37, 2 (1995), 381–394.
17. Forlizzi, J., Barley, W.C., and Seder, T. Where should I turn?: Moving from individual to collaborative navigation strategies to inform the interaction design of future navigation systems. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Atlanta, GA, Apr. 10–15). ACM Press, New York, 1261–1270.
18. Gaspar, J.G., Street, W.N., Windsor, M.B., Carbonari, R., Kaczmarek, H., Kramer, A.F., and Mathewson, K.E. Providing views of the driving scene to drivers’ conversation partners mitigates cell-phone-related distraction. *Psychological Science* 25, 12 (2014), 2136–2146.
19. Gold, C., Dambrock, D., Lorenz, L., and Bengler, K. Take over! How long does it take to get the driver back into the loop? In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (San Diego, CA, Sept. 30–Oct. 4). Human Factors and Ergonomics Society, Santa Monica, CA, 2013, 1938–1942.
20. Gomes, L. Urban jungle a tough challenge for Google autonomous cars. *MIT Technology Review* (July 24, 2014); <https://www.technologyreview.com/s/529466/urban-jungle-a-tough-challenge-for-googles-autonomous-cars/>
21. He, J., Becic, E., Lee, Y.C., and McCarter, J.S. Mind wandering behind the wheel: Performance and oculomotor correlates. *Human Factors* 53, 1 (2011), 13–21.
22. Hoff, K.A. and Bashir, M. Trust in automation: Integrating empirical evidence on factors that influence trust. *Human Factors* 57, 3 (2014), 407–434.
23. Itoh, M. and Inagaki, T. Design and evaluation of steering protection for avoiding collisions during a lane change. *Ergonomics* 57, 3 (2013), 361–373.
24. Kun, A.L. and Medenica, Z. Video call, or not, that is the question. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Austin, TX, May 5–10). ACM Press, New York, 2012, 1631–1636.
25. Lee, J.D., McGehee, D.V., Brown, T.L., and Reyes, M.L. Collision warning timing, driver distraction, and driver response to imminent rear-end collisions in a high-fidelity driving simulator. *Human Factors* 44, 2 (2002), 314–334.
26. Leshed, G., Velden, T., Rieger, O., Kot, B., and Sengers, P. In-car GPS navigation: Engagement with and disengagement from the environment. In *Proceedings of the ACM Conference on Human Factors in Computing Systems* (Florence, Italy, Apr. 5–10). ACM Press, New York, 2008, 1675–1684.
27. NHTSA. *Estimating Lives Saved by Electronic Stability Control, 2008–2012*. DOT HS 812 042. National Highway Traffic Safety Administration, Washington, D.C., 2014.
28. Norman, D.A. The problem of automation: Inappropriate feedback and interaction, not ‘over-automation.’ *Philosophical Transactions of the Royal Society B* 327, 1241 (Apr. 1990), 585–593.
29. Palmer, E.A., Hutchins, E.L., Ritter, R.D., and VanCleave, I. *Altitude Deviations: Breakdowns of an Error-Tolerant System*. NASA Technical Memorandum NASA TM-108788. NASA, Moffett Field, CA, 1993; <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19940011077.pdf>
30. Perrow, C.B. *Normal Accidents: Living with High-Risk Technologies*. Basic Books, New York, 1984.
31. Roy, M.M. and Liersch, M.J. I am a better driver than you think: Examining self-enhancement for driving ability. *Journal of Applied Social Psychology* 43, 8 (2014), 1648–1659.
32. Rueda-Domingo, T., Lardelli-Claret, P., Luna-del-Castillo, J.D., Jiménez-Moleón, J.J., García-Martín, M., and Bueno-Cavanillas, A. The influence of passengers on the risk of the driver causing a car collision in Spain: Analysis of collisions from 1990 to 1999. *Accident Analysis & Prevention* 36, 3 (2004), 481–489.
33. Sarter, N.B., Woods, D.D., and Billings, C.E. Automation surprises. In *Handbook of Human Factors & Ergonomics, Second Edition*, G. Salvendy, Ed. John Wiley & Sons, Inc., New York, 1997.
34. Sarter, N.B., Mumaw, R.J., and Wickens, C.D. Pilots’ monitoring strategies and performance on automated flight decks: An empirical study combining behavioral and eye-tracking data. *Human Factors* 49, 3 (2007), 347–357.
35. Stanton, N.A. and Young, M.S. Driver behavior with adaptive cruise control. *Ergonomics* 48, 10 (2005), 1294–1313.
36. Strayer, D.L., Cooper, J.M., Turrill, J., Coleman, J., Medeiros-Ward, N., and Biondi, F. *Measuring Cognitive Distraction in the Automobile*, Monograph 01483811. AAA Foundation for Traffic Safety, Washington, D.C., 2013; <https://www.aaafoundation.org/sites/default/files/MeasuringCognitiveDistractions.pdf>
37. Vollrath, M., Meilinger, T., and Kruger, H.-P. How the presence of passengers influences the risk of a collision with another vehicle. *Accident Analysis & Prevention* 34, 5 (Sept. 2002), 649–654.
38. Wiener, E.L. *Human Factors of Cockpit Automation: A Field Study of Flight Crew Transition*. NASA Contractor Report 177333. NASA, Moffett Field, CA, 1985; <http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19850021625.pdf>
39. Wiener, E.L. Cockpit automation. In *Human Factors in Aviation*, E.L. Wiener and D. Nagel, Eds. Academic Press, San Diego, CA, 1988, 433–461.
40. Wierwille, W.W. An initial model of visual sampling of in-car displays and controls. In *Vision in Vehicles IV*, A.G. Gale et al., Eds. Elsevier Science Publishers, Amsterdam, the Netherlands, 1993, 271–279.
41. Winsor, M. Woman accused of using Facebook while driving is charged with homicide. *CNN* (Sept. 6, 2014); <http://www.cnn.com/2014/09/05/justice/north-dakota-cellphone-crash/>
42. Yanko, M.R. and Spalek, T.M. Driving with the wandering mind: The effect that mind wandering has on driving performance. *Human Factors* 56, 2 (2014), 260–269.
43. Yanko, M.R. and Spalek, T.M. Route familiarity breeds inattention: A driving simulator study. *Accident Analysis & Prevention* 57 (Aug. 2013), 80–86.
44. Young, M.S. and Stanton, N.A. Back to the future: Brake reaction times for manual and automated vehicles. *Ergonomics* 50, 1 (2007), 46–58.

Stephen M. Casner (stephen.casner@nasa.gov) is a research psychologist in the Human Systems Integration Division at NASA Ames Research Center, Moffett Field, CA.

Edwin L. Hutchins (ehutchins@ucsd.edu) is an emeritus professor of cognitive science at the University of California, San Diego.

Don Norman (dnorman@ucsd.edu) is a Fellow of the ACM and director of the Design Lab at the University of California, San Diego; his books *Design of Everyday Things* and *Design of Future Things* address many of the issues raised in this article.

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

DOI:10.1145/2901919

Data-centric abstractions and execution strategies are needed to exploit parallelism in large-scale graph analytics.

BY ANDREW LENHARTH, DONALD NGUYEN, AND KESHAV PINGALI

Parallel Graph Analytics

THE STUDY OF graphs began in the 18th century with Euler's work on the "Bridges of Königsberg" problem, but it is only since the creation of the Web two decades ago that large-scale parallel computers have been used to study graph properties. Today, this field, known as "parallel graph analytics," touches all our lives, even if we are not aware of it. When we use a Web search engine or get a friend recommendation on Facebook, graph analytics is at work behind the scenes. Companies like Amazon and Netflix use parallel graph analytics to find patterns of purchases by their customers, enabling them to make targeted product recommendations. And intelligence agencies use parallel graph analytics to find key players, called "centralities," in terrorist networks.

Using computers to study graph properties is not new; 40 years ago, in 1976, mathematicians Kenneth Appel and Wolfgang Haken ran a FORTRAN program for 1,200 hours to prove the four-color theorem for planar graphs.^a Today, we are interested in studying not only the mathematical properties of different

kinds of graphs (such as planar graphs) but also in computing and exploiting properties of particular graphs that arise from big datasets. There are three main challenges in performing computations on these graphs—large graph sizes, diversity in graph structure, and the complex patterns of parallelism in graph analytics applications. Most current parallel abstractions and implementations fail to provide solutions for at least one of them.

Large graph sizes. In today's era of big data, parallel graph analytics systems must deal with graphs with billions of nodes and trillions of edges; for example, Web search engines are based on the indexable Web graph,¹⁹ which has more than 50 billion nodes, representing webpages, and one trillion edges, representing hyperlinks between pages. The Facebook graph has approximately one billion nodes, representing Facebook users, and 200 billion edges, representing friendships between users.³⁶ Parallel computing is essential for analyzing properties of graphs of such size in reasonable time.

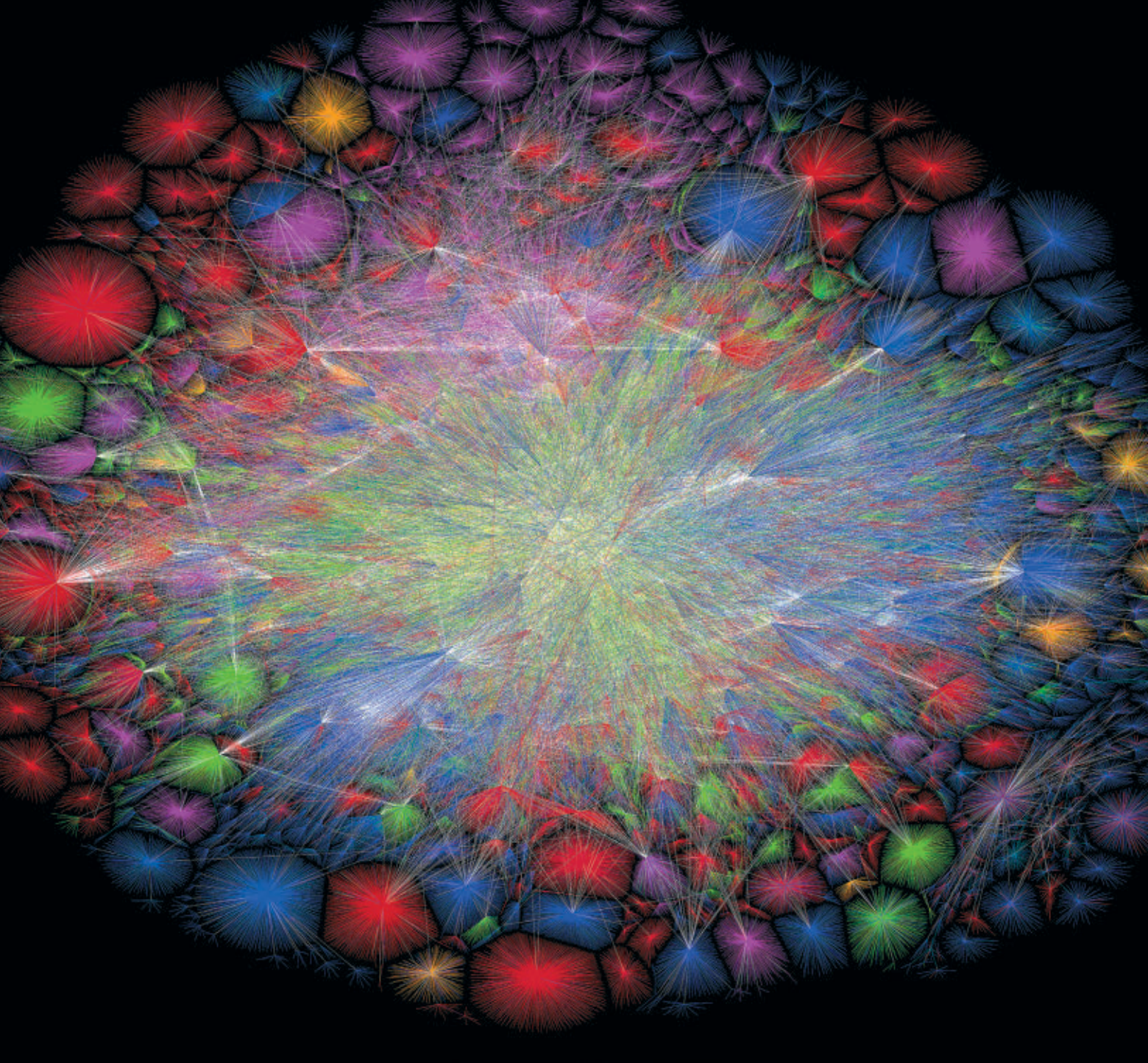
Diversity in graph structure. Graph structure is very diverse and has a major impact on the performance of graph analytics algorithms.

Planar graphs and graphs (such as road networks) have the property that the number of edges connected to a node, or the "degree" of that node, is fairly uniform across all nodes and is a small number that does not depend on the size of the graph. As a consequence, the average length of the shortest path between pairs of nodes, or the "average diameter of the graph," increases rapidly as we consider graphs of increasing size.

» key insights

- Parallelism in graph algorithms is qualitatively different from parallelism in high-performance computing algorithms.
- Data-centric abstractions are key to understanding parallelism in graph analytics applications.
- Different parallel execution strategies are needed for different types of graphs.

^a A graph is planar if it can be drawn on a sheet of paper so its edges do not cross each other.



Visualization of the routing paths of the Internet.

Studies have shown social network graphs like the Web and Facebook graphs have a very different structure from planar graphs; although the average degree of a node is small, the average diameter of randomly generated social network graphs grows only as the logarithm of the number of nodes.^{7,21} To understand how this can happen, consider an airline route map. Most airlines organize flights on the hub-and-spoke model; a small number of cities are chosen as hubs and connected to many other cities by direct flights, but most cities have

only a small number of flights, usually only to one hub. Nevertheless, you can travel between most pairs of cities by taking just three flights—a direct flight to the hub that serves your starting city, a flight from there to a hub that serves your destination, and then a flight on to your destination.

Like airline route maps, social network graphs have a “lumpy” node structure, known formally as a “power-law structure,” in which a few richly connected hubs can be exploited to traverse the network in a small number of hops; for example, the average diam-

eter of the Facebook graph is approximately five, even though the graph has more than one billion nodes.³⁶

A third category of graphs, called “random graphs” or “Erdős-Rényi graphs,” is useful for studying certain combinatorics problems. One way to construct random graphs is to start with some number of nodes and then add edges incrementally between randomly selected pairs of nodes. In contrast, power-law graphs arise from a “rich get richer” generative model in which the selection is biased in favor of nodes with higher degrees.

For many analytics problems, different algorithms must be used for uniform-degree, random, and power-law graphs to obtain good parallel performance, as we show here.

Need for new parallel abstractions. Most existing abstractions and implementations for parallel computing were developed for computational science applications in which the main parallelism pattern is data parallelism; an example of a data-parallel operation is “map,” which applies a function to each element of a set, producing a new set. Systems like Hadoop enable programmers to express and exploit data parallelism without having to write low-level parallel code.

Some graph analytics algorithms have data parallelism, but others exhibit a more complex parallelism pattern called “amorphous” data-parallelism,²⁹ described later in this article. Unlike in data-parallel programs, tasks in amorphous data-parallel programs may or may not be able to run

in parallel, depending on the graph structure and values known only at runtime. To exploit this parallelism pattern, systems need to find opportunities for parallel execution while executing the program in parallel. As a consequence, abstractions and implementations for data-parallelism are not adequate for many parallel graph analytics algorithms.

Landscape of Graph Analytics Algorithms

To understand the patterns of parallelism in graph analytics algorithms, it is convenient to use concepts from the “operator formulation of algorithms,”²⁹ a data-centric abstraction of algorithms shown pictorially in Figure 1. To illustrate these concepts, we use the Dijkstra and Bellman-Ford algorithms for the single-source shortest-path problem (SSSP).⁵ Given a weighted, undirected graph $G = (V, E, w)$, where V is the set of nodes, E is the set of edges, and w is a map from edges to positive edge weights, the SSSP problem is to compute the length of the shortest path from a given source node s to every other node. Both algorithms are described in detail in the next section.

The operator formulation has a local view and a global view of algorithms, as summarized in Figure 2.

The local view is described by an operator, which is a graph update rule applied to an active node in the graph; some algorithms have active edges, but, here, we refer only to active nodes. Each operator application, called an “activity,” reads and writes a small region of the graph, called the “neighborhood” of that activity, around the active

node. In Dijkstra’s algorithm, the operator, called the “relaxation” operator, uses the label of the active node to update the labels of its neighbors. Figure 1 shows active nodes as filled dots and neighborhoods as clouds surrounding active nodes for a generic algorithm. An active node becomes inactive when the activity is completed.

In general, operators can modify the graph structure of the neighborhood by adding and removing nodes and edges; they are called “morph” operators. In most graph analytics applications, operators update only labels on nodes and edges without changing the graph structure. These operators are called “local computation” operators; a “pull-style” operator reads the labels of nodes in its neighborhood and writes to the label of its active node, while a “push-style” operator reads the label of the active node and writes the labels of other nodes in its neighborhood. Dijkstra’s algorithm uses a push-style operator. In algorithms that operate on several data structures, some data structures may be read-only, in which case the operator is a “reader” for those data structures.

Neighborhoods can be distinct from the set of immediate neighbors of an active node, and can, in principle, encompass the entire graph, although usually they are small regions of the graph surrounding the active node. Neighborhoods of different activities can overlap; in Figure 1, node n is contained in the neighborhoods of both activities A and B . In a parallel implementation, the semantics of reads and writes to such overlapping regions, or the “memory model,” must be specified carefully.

The global view of the algorithm is described by the locations of active nodes in the graph and the order in which they must appear to have been ved by the implementation.

In “topology-driven” algorithms, the locations of active nodes are determined by the graph structure. These algorithms make a number of sweeps over the graph; in each such sweep, all nodes are initially active,^b and the

^b In some algorithms, a subset of nodes (such as interior nodes in a mesh) are initially active in each sweep, but this subset is determined by the graph topology.

Figure 1. Operator formulation.

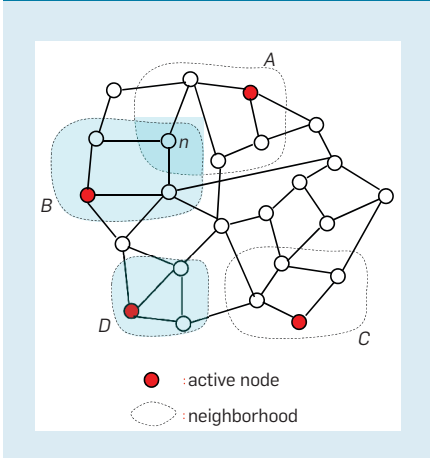
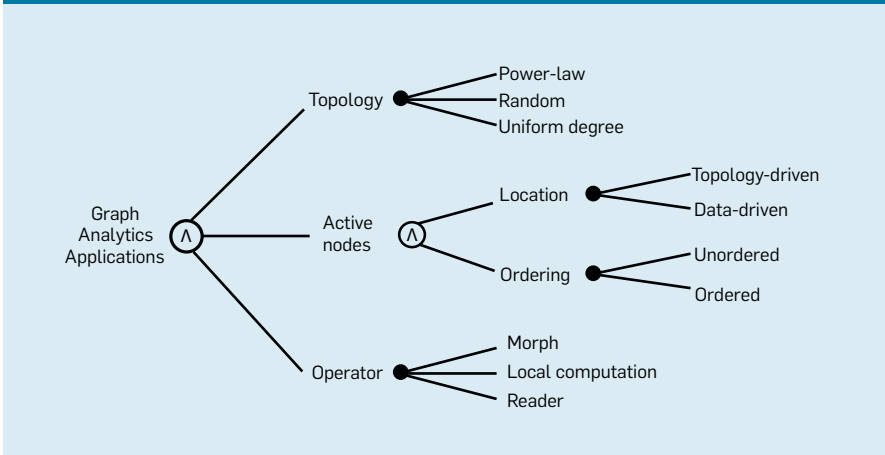


Figure 2. TAO analysis of graph applications.



sweeps terminate when some global quiescence condition is reached; the Bellman-Ford algorithm is an example. On the other hand, in data-driven algorithms, there is an initial set of active nodes, and the execution of activities may cause other nodes to become active on the fly. These algorithms do not have any algorithmic notion of sweeps, terminating when there are no more active nodes in the graph. Dijkstra's algorithm is an example; only the source node is active initially and other nodes become active as a side effect of graph updates.

The second dimension of the global view of algorithms is "ordering." In "unordered" algorithms, any order of processing active nodes is semantically correct; each sweep of the Bellman-Ford algorithm is an example. Some orders may be more efficient than others, so unordered algorithms sometimes assign soft priorities to activities, but these priorities are only suggestions to the runtime system, and priority inversions are permitted in the execution. In contrast, "ordered" algorithms requiring active nodes appear to have been processed in a specific order; Dijkstra's algorithm is an example. This order is specified by assigning priorities to active nodes, and the implementation is required to process active nodes so they appear to have been scheduled for execution in strict priority order, from earliest to latest.

Like SSSP, many graph analytics problems can be solved by both topology-driven and data-driven algorithms, but each type of algorithm makes different trade-offs. Topology-driven algorithms are easier to implement because graph representations usually store the nodes of the graph in an array or a list, so iteration over active nodes can be implemented simply as iteration over this data structure. In contrast, data-driven algorithms must use a workset (or multiset) to hold the active nodes. Sequential implementations usually use lists and priority queues to implement worksets for unordered and ordered algorithms, respectively. Nguyen et al.²⁷ described a scalable implementation of a parallel workset supporting soft priorities.

Data-driven algorithms can be more work efficient. Topology-driven algorithms visit nodes even if there is no



Unlike in data-parallel programs, tasks in amorphous data-parallel programs may or may not be able to run in parallel, depending on the graph structure and values known only at runtime.



work to do there, whereas data-driven algorithms can focus on regions of the graph where useful work needs to be done, making them asymptotically faster than their topology-driven counterparts for many problems; for example, the asymptotic complexity of the Bellman-Ford algorithm is $O(|E||V|)$, whereas the complexity of Dijkstra's algorithm is $O(|E| \log(|V|))$.

Figure 2 is a graphical presentation of these categories of graph applications. We call it TAO analysis for its three main dimensions—Topology of the input graph, Active nodes location and ordering, and Operator. There is no explicit notion of parallelism in TAO analysis or a distinction between sequential and parallel algorithms.

These concepts map directly to programming constructs in object-oriented languages. Graphs are implemented using abstract data types that support a common API but represent the graph in memory in different schemes. Iteration over active nodes can be expressed succinctly by unordered and ordered set iterators that permit new elements to be added to the set during the iteration.²⁹ For topology-driven algorithms, this iterator is nested within an ordinary (ordered) loop that iterates until some convergence criterion is reached. The body of the iterator is an implementation of the operator and makes calls to a graph API for accessing graph elements. Some algorithms have multiple phases, and each phase may use a different operator; other algorithms may require the interleaved execution of several operators.

Illustrative Examples

We illustrate these ideas through algorithms for SSSP and collaborative filtering.

There are many algorithms for SSSP, including chaotic relaxation, Dijkstra, delta-stepping, and Bellman-Ford.^{5,25} In the literature, they appear to be unrelated to each other, but TAO analysis reveals their similarities and differences. Each node v is given a distance label $dist(v)$ that maintains the length of the shortest known path to that node from the source. This field is initialized to 0 at the source s and ∞ at all other nodes. The relaxation operator in SSSP algorithms is defined as follows: if (u, v) is an edge and $dist(u) + w(u, v) < dist(v)$,

the value of $dist(v)$ is updated to $dist(u) + w(u, v)$. An active node is relaxed by applying this operator to all the edges connected to it. This is a push-style operator, though it is also possible to implement a pull-style operator. Each relaxation may lower the $dist$ label of some node, and when no further lowering can be performed anywhere in the graph, the resulting node labels are the shortest distances from the source to the nodes, regardless of the order in which the relaxations were performed. Breadth-first numbering of a graph is a special case of SSSP in which all edge labels are 1.

Differences between SSSP algorithms arise in the active nodes dimension of TAO analysis.

“Chaotic relaxation”³ is a data-driven, unordered algorithm. The workset initially contains only the source node (for the push-style operator), and active nodes are chosen randomly by the algorithm for execution from the workset. This algorithm exposes a lot of parallelism for most input graphs, but the number of relaxations may be exponential in the graph size, so the algorithm is not work efficient.

Dijkstra’s algorithm is a data-driven, ordered algorithm with a push-style operator. The workset initially contains only the source node. Active nodes are scheduled in strictly non-decreasing order of node labels, using a priority queue in which the key is the node label. This algorithm is work efficient but may not expose much parallelism.

“Asynchronous delta-stepping”^c is a compromise between the extremes of chaotic relaxation and Dijkstra’s algorithm and can be viewed as controlled chaotic relaxation. The workset is implemented as a sequence of buckets, and an active node with label d is mapped to the bucket at position d / Δ , where Δ is a user-defined parameter. Nodes within a given bucket can be relaxed in any order, but buckets are processed in strictly increasing order. If Δ is 1, the algorithm behaves like Dijkstra’s algorithm, and if $\Delta = \infty$, the algorithm behaves like chaotic relaxation.

^c This is a version of the delta-stepping algorithm of Meyer et al.²⁵ that does not require preprocessing; PHAST is another SSSP algorithm for high-diameter graphs (such as road networks) that quickly preprocess the graph to answer SSSP queries.⁶



For many analytics problems, different algorithms must be used for uniform-degree, random, and power-law graphs to obtain good parallel performance.



In social network graphs, a small value of Δ is preferable, as this choice promotes work efficiency without choking off parallelism;^d for uniform-degree graphs, a larger-value Δ must be used to expose parallelism.

Bellman-Ford is a topology-driven algorithm that makes $O(|V|)$ unordered sweeps over the graph. Most implementations use a push-based operator.

In conventional Bellman-Ford, updates to a node label in a sweep are available immediately to other activities in that sweep; such sweeps are Gauss-Seidel iterations in the linear algebra sense. “Jacobi-style” Bellman-Ford maintains two distance labels—oldDist and newDist—on each node. In each sweep, a pull-style operator is applied to an active node to read the oldDist values from its neighbors and update the newDist label of that active node. The roles of oldDist and newDist are reversed in the next iteration.

The computations performed by each sweep in Jacobi-style Bellman-Ford can be expressed succinctly in terms of a generalized matrix-vector product, where the matrix is the incidence matrix of the graph, and the vector gathers node labels—either oldDist or newDist—into a single data structure. Figure 3 shows a directed graph and its incidence matrix A . Each node has labels named x and y , and these labels can be collected to form the vectors x and y . If there is an edge (i, j) in the graph, j is said to be an “out-neighbor” of i , and i is said to be an “in-neighbor” of j . The matrix computation $y = y + Ax$ can be interpreted graphically as a topology-driven computation in which for each node i and for each of its out-neighbors j , the product of the weight on edge (i, j) and the value of the x label of j is added to the y label of i . The matrix computation $y = y + A^T x$ with the transpose of the incidence matrix represents a similar computation with the in-neighbors of each node i .

Standard linear algebra uses arithmetic operations over the field $(\mathbb{R}, +, \times)$. The generalized matrix-vector product for the SSSP problem uses operations over the algebraic semi-ring $(\mathbb{R}, \min, +)$ and computes $A^T x$.

^d Strictly speaking, this is true if all edge labels are approximately equal in value.

Collaborative filtering. Given an incomplete database of ratings given by users to items, the goal is to predict how these users would rate items they have not yet rated. One way to solve this problem is through low-rank approximation. The database of ratings is represented as a matrix R , and low-rank approximation tries to find two low-dimensional dense matrices W and H such that $R \approx WH$, as shown in Figure 4.

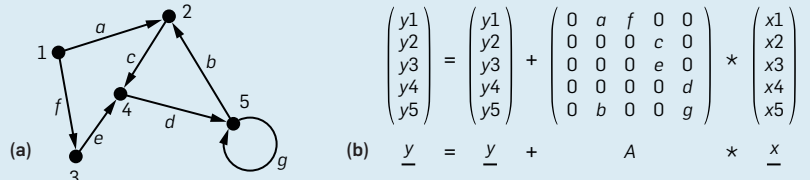
Low-rank approximation can be formulated as a graph problem. The database is represented as a bipartite graph between users and items; if user u gave a rating r to an item v , there is an edge (u,v) in the graph with weight r . The matrices W and H are represented by vectors of length k associated with the nodes representing users and items, respectively, as shown in Figure 4. One way to compute these vectors is to use an iterative algorithm like stochastic gradient descent SGD, which initializes these vectors to some values and then makes a number of sweeps over the bipartite graph until a convergence test is satisfied. In each sweep, all edges (u,v) are visited in some order. If the inner product of the vectors on nodes u and v is not equal to the weight on edge (u,v) , the difference is used to update the two vectors. This is a topology-driven algorithm in which edges are active.

Parallel Graph Analytics

Parallelism can be exploited by processing active nodes in parallel, subject to neighborhood and ordering constraints. Since neighborhoods can overlap, the memory model, which defines the semantics of reads and writes in overlapped regions, may prevent some activities with overlapping neighborhoods from executing in parallel. In addition, ordering constraints between activities must be enforced. We call this pattern of parallelism “amorphous” data-parallelism;²⁹ it is a generalization of data-parallelism in which there may be neighborhood and ordering constraints that prevent all activities from executing in parallel, and the execution of an activity may create new activities.

Figure 5 outlines the important choices in implementing parallel graph analytics programs. There are

Figure 3. Graphical interpretation of generalized MVM.



two popular memory models: bulk-synchronous parallel (BSP)-style semantics³⁷ and transactional semantics.

BSP-style semantics. The program is executed in rounds (also known as “super-steps”), with barrier synchronization between rounds. Writes to the graph are considered communication from a round to the following round, so are applied to the graph only at the beginning of the following round. Multiple updates to a label are resolved as in the parallel random access machine models (such as by using a reduction operation to combine the updates into a single update).¹³ Each round thus consists of updating the graph based on values from the previous round, performing activities, and sending values to the next round.

As an example, consider the execution of the Bellman-Ford algorithm under BSP-style semantics. Each sweep over the graph is performed in one round. In each such round, the operator is applied to all nodes in parallel, but the updates to node labels are buffered and applied to the graph at the beginning of the next round. Multiple updates to the same node are combined by computing their minimum. Note under these semantics, the Bellman-Ford algorithm behaves like the Jacobi-style Bellman-Ford algorithm discussed earlier; updates made in a round become available to other relaxations only in the next round.

To reduce the overheads of data transfers between rounds, practical implementations of BSP-style semantics, particularly on distributed-memory machines, use a more coarse-grain approach in which tasks are associated not with individual activities but with logical or physical partitions of the graph. A task executes all activities whose active nodes lie in its partition; updates within the partition are performed in place and are available in

Figure 4. Collaborative filtering.

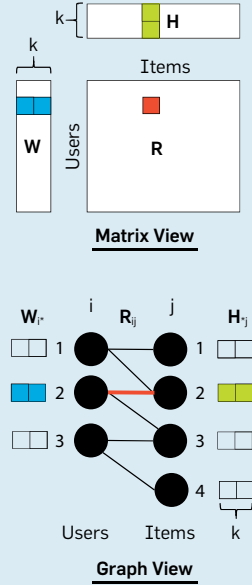
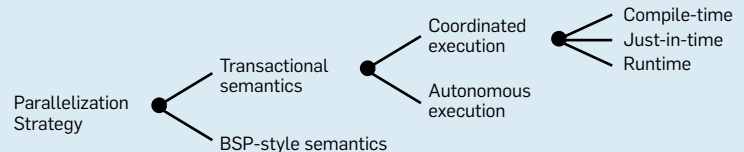


Figure 5. Parallelization strategies.



the same round to other activities, but updates to remote nodes are buffered and applied at the beginning of each super-step.

BSP-style parallelization may work well for graph analytics applications in which the number of activities in each round is large enough to keep the processors of the parallel machine busy; one example is breadth-first search (BFS) on a power-law graph. Each such round handles nodes at a single BFS level and computes labels for nodes at the next BFS level. Since the average diameter of power-law graphs is small, most rounds involve many parallel activities.

On the other hand, BSP-style parallelization may not perform well for graphs with high average diameter (such as road networks and meshes). This is because the number of super-steps required to execute the algorithm may be large, and the number of activities in each super-step may be small, as discussed later.

Transactional semantics. In this model, parallel execution of activities is required to produce the same answer as executing activities one at a time in some order that respects priorities. This means activities should not “see” concurrently executing activities, and the updates made by an activity should become visible to other activities only

after that activity completes execution. Formally, these two properties of transactional execution are known as “isolation” and “atomicity.”¹¹

Transactional semantics are implemented by preventing activities from executing in parallel if they conflict. For unordered algorithms, a conservative definition is that activities conflict if their neighborhoods overlap. In Figure 1, activities *A* and *B* conflict because node *n* is in both their neighborhoods. Activities *C* and *D* do not conflict and can be executed in parallel with *A* or *B*. It is possible to use more relaxed definitions like commutativity checking, which exploits the fact that many graph analytics algorithms have commutative updates that can be implemented through atomics, thus eliding full transactional semantics.¹⁸

Given a definition of conflicts, the implementation must ensure conflicting activities do not update the graph in parallel. This can be accomplished through “autonomous scheduling” or “coordinated scheduling.”

In autonomous scheduling, activities are executed speculatively.³¹ If a conflict is detected with other concurrently executing activities, some activities are aborted, enabling others to make progress; otherwise, the activity commits, and its updates become visible to other activities.

Coordinated scheduling strategies ensure conflicting activities do not execute simultaneously.²⁹ For some algorithms (such as those that can be expressed through a generalized sparse matrix-vector product), static analysis of the operator shows active nodes can be executed in parallel without any conflict checking. This is called

“static parallelization” and is similar to auto-parallelization of dense array programs. Just-in-time parallelization pre-processes the input graph to find conflict-free schedules (such as through graph coloring); this technique can be used for topology-driven algorithms like Bellman-Ford in which neighborhoods are independent of data values. Runtime parallelization is general; the algorithm is executed in a series of rounds, and in each round, a set of active nodes is chosen, their neighborhoods are computed, and a set of non-conflicting activities are selected and executed. This approach has been used for deterministic execution of unordered algorithms.²⁸

Systems for Graph Analytics

Systems for graph analytics usually support a subset of the algorithm classes and parallelization strategies discussed earlier. Here, we restrict ourselves to describing only a few of the many interesting systems for parallel graph analytics in the literature.

Most systems support only “vertex programs,” an abstraction introduced by the Pregel system;²³ in the operator formulation, vertex programs are local computation operators in which the neighborhood is restricted to be the active node and its immediate neighbors. Vertex programs have limited expressiveness because important graph operations like “pointer-jumping,” as in Figure 6, cannot be expressed as vertex programs. Efficient parallel algorithms for problems like connected components use extended versions of pointer-jumping that are similar to the `find` operation in disjoint-set union-find

Figure 6. With pointer-jumping, *m*’s parent pointer is redirected from parent *p* to grandparent *g*.

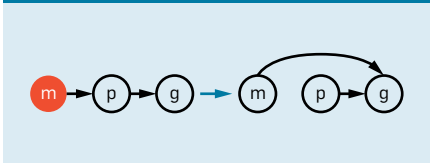
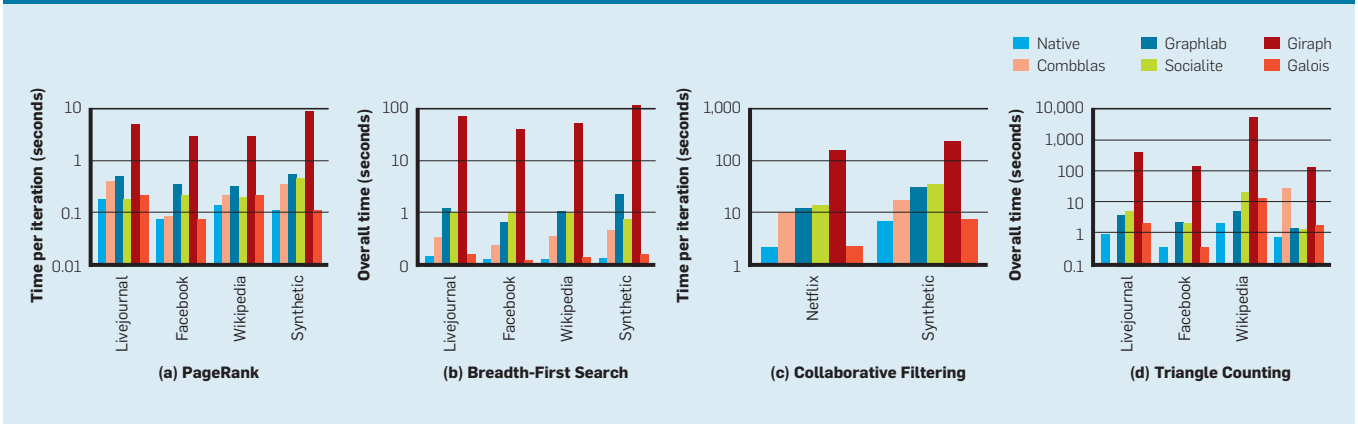


Figure 7. Comparison of native implementations with graph analytics frameworks using shared-memory from Satish et al.³²



in disjoint-set union-find.^{13,14} Programming systems that do not support these types of operators may require inefficient algorithms for some problems.

Since very large graph datasets may not fit into the memory of a single host, one solution is to partition the graph among the hosts of a distributed-memory cluster. Some systems partition graph nodes, while others partition graph edges. A node partition, also known as a “1-D partition,” assigns each node to one host; most graph partitioners try to partition nodes more or less equally among hosts while minimizing the number of edges that span multiple hosts. This works well for topology-driven algorithms on uniform-degree graphs since all nodes are active and the amount of computation is roughly the same at all nodes. However, node partitioning may lead to computation and communication imbalance for power-law graphs due to the lumpy nature of node degrees in these graphs. One solution is to partition edges among hosts, an approach sometimes called “2-D partitioning” because it is analogous to partitioning the matrix representation of the graph into 2D blocks and assigning each block to a host.

Examples of distributed-memory graph analytics systems include the following:

CombBLAS. CombBLAS⁴ pioneered the generalized sparse matrix operations approach to formulating graph algorithms and is well suited for unordered, topology-driven algorithms with pull-style operators like Jacobi-style Bellman-Ford; CombBLAS supports 2-D partitions;

Pregel. Pregel²³ introduced the concept of vertex programs and supports a limited form of morph operators; Pregel’s parallelization strategy uses BSP-style semantics and supports 1-D partitions;

Giraph. Giraph⁸ is an implementation of Pregel from Facebook and implemented on top of the Hadoop platform for fault tolerance; and

PowerGraph. PowerGraph⁹ is a distributed-memory programming model for vertex programs with BSP-style semantics and supports an implementation of 2D partitioning called “vertex

cuts”; GraphLab²⁰ is an earlier shared-memory version of PowerGraph.

Examples of shared-memory systems include the following:

Ligra. Ligra³³ is a shared-memory programming model for vertex programs with BSP semantics and can switch automatically between push-style and pull-style operators during execution of an algorithm, which is useful for some algorithms on power-law graphs; and

Galois. Galois²⁷ is a general shared-memory programming model, implemented in sequential C++, and, unlike the systems outlined here, is not restricted to graph analytics; it supports morph operators and operators with arbitrary neighborhoods and implements transactional semantics with both autonomous and coordinated runtime scheduling.

Performance Studies

Graph analytics systems make the job of the application programmer easier through abstraction but may also impose limitations; for example, the best

algorithm for a problem may not be expressible in the abstractions supported by a given system. Moreover, abstraction may introduce a performance penalty compared to explicitly parallel constructs like pthreads and MPI. This section describes performance studies that shed light on these matters.

Power-law graphs. A 2014 Intel study³² considered four applications: PageRank, BFS, triangle counting, and collaborative filtering. The researchers wrote programs for each application in several frameworks, including CombBLAS, Graphlab, Giraph, and Galois. As a baseline for performance comparisons, the researchers also wrote parallel programs for each application in a “native” parallel notation—either pthreads or OpenMP.

We described BFS and collaborative filtering earlier in this article. PageRank is a topology-driven algorithm with a push-style operator on directed graphs, computing the relative importance of nodes in a graph as a function of its incoming neighbors. Triangle counting is a simple example of finding

Figure 8. Comparison of native implementations with graph analytics frameworks using distributed-memory from Satish et al.³²

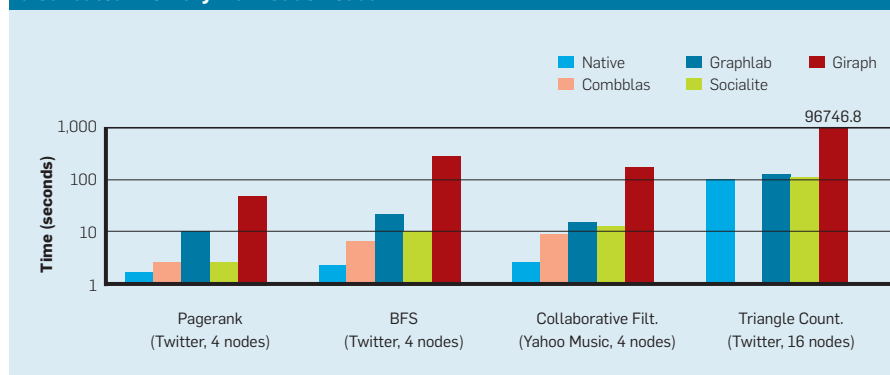
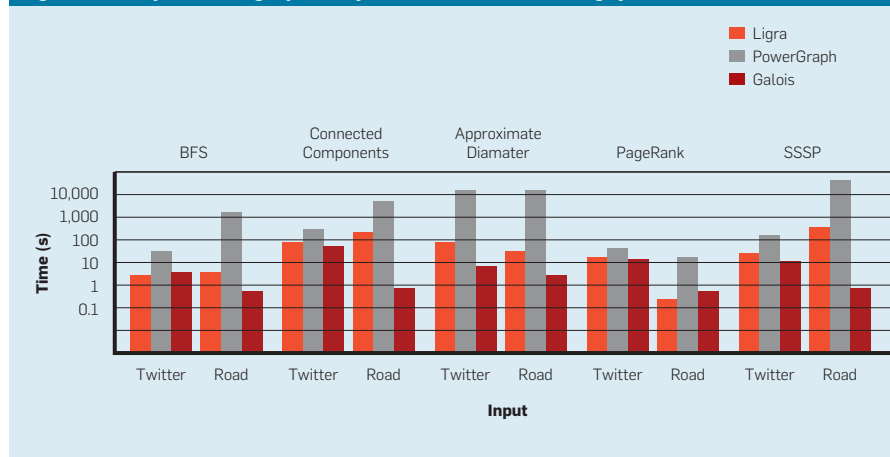


Figure 9. Comparison of graph analytics frameworks from Nguyen et al.²⁷



motifs in graphs. A triangle is formed in an undirected graph whenever two nodes have a common neighbor. The problem is to count the number of triangles in a given undirected graph.

Figure 7 shows the running times of these programs on several power-law graphs on a 24-core Intel Xeon CPU E5-2697 with 64GB of DRAM. Note the y-axis is a log-scale. The Facebook graph has three million nodes and 42 million edges. The synthetic graph has 537 million nodes and 8.6 billion edges. For collaborative filtering, a synthetic bipartite graph included 63 million users, 1.3 million items, and 17 billion ratings. PageRank times are for one iteration.

While the native parallel programs performed best, Galois programs are close in performance; the Intel study reported they are slower by a factor of 1.1–1.2 for PageRank, BFS, and collaborative filtering and 2.5X for triangle counting. Giraph is 100–1,000 times slower than native code due to I/O in the Hadoop system. While PageRank and BFS fit the generalized sparse matrix vector multiply pattern directly and are thus easy to express in CombBLAS, triangle counting is more tricky. The implementation of triangle counting in the Intel study³² computed the matrix product A^2 and ran out of memory; Azad et al.¹ described a better CombBLAS algorithm.

Figure 8 shows performance numbers from the Intel study³² for a four-host distributed-memory system; there are no performance bars for Galois, since it is shared-memory, although a distributed-memory version will be available in 2016. In this

experiment, native parallel programs were written with a combination of MPI and OpenMP/threads. As in the shared-memory case, Giraph is substantially slower than the other systems due to I/O. CombBLAS and Socialite perform comparably and are faster than GraphLab, which is limited by network bottlenecks.³²

Influence of graph structure. Nguyen et al.²⁷ studied the interaction between graph structure and parallelization strategy. They evaluated five problems in PowerGraph, Ligra, and Galois—BFS, connected components, approximate diameter, PageRank, and SSSP. Their experiments measured running times on a 40-core Intel E7-4860 with 128GB of memory for two inputs graphs—a Twitter graph (51 million nodes, 2.0 billion edges) and a U.S. road network (24 million nodes, 58 million edges). The Twitter graph is a power-law graph, whereas the road network is a uniform-degree graph smaller than the Twitter graph but with a much larger average diameter. Figure 9 shows these running times; note the y-axis is a log-scale.

In the BSP-style frameworks—Ligra and PowerGraph—BFS and SSSP take longer on the road network than on the Twitter graph, even though the road network is almost two orders of magnitude smaller than the Twitter graph. The reason for the difference is BSP-style parallelization does not expose much parallelism if the number of activities in each round is small and the number of rounds is large, as is the case with road networks and other uniform-

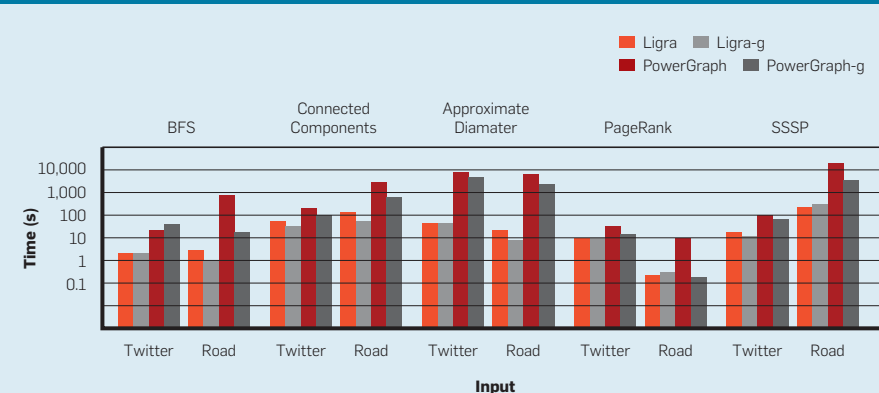
degree graphs. For SSSP, Ligra, and PowerGraph, implementations use the Bellman-Ford algorithm but filter nodes so each sweep considers only those nodes with labels updated by the previous sweep. In contrast, the Galois implementations can use data-driven algorithms like asynchronous delta-stepping, and the asynchronous, priority-guided execution of activities in the Galois system adapts to both the power-law graph and the road network.

The performance numbers for connected components highlight the limitations of vertex programs. The Galois implementation uses an efficient algorithm based on pointer-jumping that cannot be expressed in frameworks limited to vertex programs, as described earlier. Ligra and PowerGraph use a label-propagation algorithm; nodes are given integer labels, and the node with the smallest label in a component propagates its label to all other nodes in that component. When the graph diameter is small, as it is for the Twitter graph, label propagation works fine, but for the road network, it is two or three orders of magnitude slower than the asynchronous pointer-jumping algorithm in Galois.

Overall, there is a wide variation in running times across different programming models solving the same problem. The variation is least for PageRank, which uses a topology-driven algorithm in all these systems. For the other problems, the performance differences between programming models can be several orders of magnitude and are quite stark for the road input, for which the large diameter heavily penalizes BSP-style parallelization and favors asynchronous, data-driven execution of activities.

Since the programming models supported by graph DSLs (such as PowerGraph and Ligra) are restrictions of the Galois programming model, they can be implemented on top of Galois through code adaptors consisting of a few hundred lines of C++ code; we call these implementations “PowerGraph-g” and “Ligra-g.” Figure 10 shows the performance of these implementations is approximately the same as the performance of the native PowerGraph and Ligra systems, showing Galois can be considered a graph analytics engine

Figure 10. Comparison of graph analytics frameworks rewritten on top of Galois from Nguyen et al.²⁷



on top of which custom graph DSLs can be layered.

Research Directions

Ongoing research in parallel graph analytics spans the gamut, from new domain-specific languages^{12,30} to novel processor architectures for supporting graph analytics^{2,16,22} and is summarized in the online appendix.

Over the past decade, the parallel computing research community has acquired a reasonably good understanding of the landscape of parallel graph analytics algorithms. Exploiting these insights to build efficiently engineered systems for parallel graph analytics, particularly on nontraditional architectures, and incorporating graph analytics into the larger ecosystem of parallel computing are the next major challenges for this new and rapidly growing research area.

Acknowledgments

We thank Jon Kleinberg (Cornell University) and Vijaya Ramachandran and Amber Hassaan (the University of Texas at Austin) for assistance with this article and the members of the Intelligent Software Systems group at the University of Texas at Austin for their contributions to the research it describes. 

References

1. Azad, A., Buluc, A., and Gilbert, J.R. Parallel triangle counting and enumeration using matrix algebra. In *Proceedings of the 29th International Parallel & Distributed Processing Symposium Workshop* (Hyderabad, India, May 25–29). IEEE International, 2015, 804–811.
2. Bader, D.A. and Madduri, K. Designing multithreaded algorithms for breadth-first search and st-connectivity on the Cray MTA-2. In *Proceedings of the 2006 International Conference on Parallel Processing* (Columbus, OH, Aug. 14–18). IEEE Computer Society Press, 2006, 523–530.
3. Bertsekas, D.P. and Tsitsiklis, J.N. *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, 1989.
4. Buluc, A. and Gilbert, J.R. The combinatorial BLAS: Design, implementation, and applications. *International Journal of High Performance Computing Applications* 25, 4 (Nov. 2011), 496–509.
5. Cormen, T., Leiserson, C., Rivest, R., and Stein, C., Eds. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.
6. Dellling, D., Goldberg, A.V., Nowatzyk, A., and Werneck, R.F. Phast: Hardware-accelerated shortest path trees. In *Proceedings of the 25th International Parallel and Distributed Processing Symposium* (Anchorage, AK, May 16–20). IEEE Computer Society Press, 2011, 921–931.
7. Fernholz, D. and Ramachandran, V. The diameter of sparse random graphs. *Random Structures & Algorithms* 31, 4 (Dec. 2007), 482–516.
8. Giraph: <http://giraph.apache.org/>
9. Gonzalez, J.E., Low, Y., Gu, H., Bickson, D., and Guestrin, C. PowerGraph: Distributed graph-parallel computation on natural graphs. In *Proceedings of the USENIX Conference on Operating Systems Design and Implementation* (Hollywood, CA, Oct. 8–10). USENIX Association, Berkeley, CA, 2012, 17–30.
10. Harish, P. and Narayanan, P.J. Accelerating large graph algorithms on the GPU using CUDA. In *Proceedings of the 14th International Conference on High Performance Computing* (Goa, India, Dec. 19–22). Springer-Verlag, Berlin, Heidelberg, 2007, 197–208.
11. Herlihy, M. and Shavit, N. *The Art of Multiprocessor Programming*. Morgan-Kaufmann, San Francisco, CA, 2012.
12. Hong, S., Chafi, H., Sedlar, E., and Olukotun, K. Green-Marl: A DSL for easy and efficient graph analysis. In *Proceedings of the 17th International Conference on Architectural Support for Programming Languages and Operating Systems* (London, U.K., Mar. 3–7). ACM Press, New York, 2012, 349–362.
13. Jaja, J. *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading, MA, 1992.
14. Karp, R.M. and Ramachandran, V. *A Survey of Parallel Algorithms for Shared-Memory Machines*. Technical Report UCB/CSD-88-408. EECS Department, University of California, Berkeley, Mar. 1988; <http://www.eecs.berkeley.edu/Pubs/TechRpts/1988/CSD-88-408.pdf>
15. Khorasani, K., Vora, K., Gupta, R., and Bhuyan, L.N. Cusha: Vertex-centric graph processing on GPUs. In *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing* (Vancouver, Canada, June 23–27). ACM Press, New York, 2014, 239–252.
16. Kim, J. and Batten, C. Accelerating irregular algorithms on GPGPUs using fine-grain hardware worklists. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, U.K., Dec. 13–17). IEEE Computer Society Press, 2014, 75–87.
17. Kulkarni, M., Burtcher, M., Casçaval, C., and Pingali, K. Lonestar: A suite of parallel irregular programs. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software* (Boston, MA, Apr. 26–28). IEEE Computer Society Press, 2009; <http://iss.ices.utexas.edu/?p=projects/galois/lonestar>
18. Kulkarni, M., Nguyen, D., Proutzoz, D., Sui, X., and Pingali, K. Exploiting the commutativity lattice. In *Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation* (San Jose, CA, June 4–8). ACM Press, New York, 2011, 542–555.
19. Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tompkins, A., and Upfal, E. The Web as a graph. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Dallas, TX, May 15–17). ACM Press, New York, 2000, 1–10.
20. Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., and Hellerstein, J.M. GraphLab: A new parallel framework for machine learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence* (Catalina Island, CA, July 8–11). Association for Uncertainty in Artificial Intelligence, 2010.
21. Lu, L. The diameter of random massive graphs. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, CA, Jan. 9–11). Society for Industrial and Applied Mathematics, Philadelphia, PA, 2001, 912–921.
22. Madduri, K., Feo, J., Cong, G., and Bader, D.A. Design of multithreaded algorithms for combinatorial problems. Chapter 31 in *Handbook of Parallel Computing: Models, Algorithms and Applications*, S. Rajasekaran and J. Reif, Eds. Chapman and Hall/CRC, Boca Raton, FL, 2007.
23. Malewicz, G., Austern, M.H., Bik, A.J., Dehnert, J.C., Horn, I., Leiser, N., and Czajkowski, G. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data* (Indianapolis, IN, June 6–11). ACM Press, New York, 2010, 135–146.
24. Merrill, D., Garland, M., and Grimshaw, A. Scalable GPU graph traversal. *SIGPLAN Notices* 47, 8 (Feb. 2012), 117–128.
25. Meyer, U. and Sanders, P. Delta-stepping: A parallel single-source shortest-path algorithm. In *Proceedings of the Sixth Annual European Symposium on Algorithms* (Venice, Italy, Aug. 24–26). Springer-Verlag, London, U.K., 1998, 393–404.
26. Nasre, R., Burtcher, M., and Pingali, K. Morph algorithms on GPUs. In *Proceedings of the 18th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Shenzhen, China, Feb. 23–27). ACM Press, New York, 2013, 147–156.
27. Nguyen, D., Lenharth, A., and K. Pingali. A lightweight infrastructure for graph analytics. In *Proceedings of the 24th ACM Symposium on Operating Systems Principles* (Farmington, PA, Nov. 3–6). ACM Press, New York, 2013, 456–471.
28. Nguyen, D., Lenharth, A., and Pingali, K. Deterministic Galois: On-demand, portable and parameterless. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems* (Salt Lake City, UT, Mar. 1–5). ACM Press, New York, 2014, 499–512.
29. Pingali, K., Nguyen, D., Kulkarni, M., Burtcher, M., Hassaan, M.A., Kaleem, R., Lee, T.-H., Lenharth, A., Manevich, R., Méndez-Lojo, M., Proutzoz, D., and Sui, X. The Tao of parallelism in algorithms. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (San Jose, CA, June 4–8). ACM Press, New York, 2011, 12–25.
30. Proutzoz, D., Manevich, R., and Pingali, K. Synthesizing parallel graph programs via automated planning. In *Proceedings of the 36th Annual SIGPLAN Conference on Programming Language Design and Implementation* (Portland, OR, June 13–17). ACM Press, New York, 2015, 533–544.
31. Rauchwerger, L. and Padua, D.A. The LRPD test: Speculative runtime parallelization of loops with privatization and reduction parallelization. *IEEE Transactions on Parallel Distributed Systems* 10, 2 (Feb. 1999), 160–180.
32. Satish, N., Sundaram, N., Patwary, M.M.A., Seo, J., Park, J., Hassaan, M.A., Sengupta, S., Yin, Z., and Dubey, P. Navigating the maze of graph analytics frameworks using massive graph datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (Snowbird, UT, June 22–27). ACM Press, New York, 2014, 979–990.
33. Shun, J. and Blöchl, G.E. Ligra: A lightweight graph-processing framework for shared memory. In *Proceedings of the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (Shenzhen, China, Feb. 23–27). ACM Press, New York, 2013, 135–146.
34. Shun, J., Blöchl, G.E., Fineman, J.T., Gibbons, P.B., Kyrola, A., Simhadri, H.V., and Tangwongsan, K. Brief announcement: The problem-based benchmark suite. In *Proceedings of the 24th Annual ACM Symposium on Parallelism in Algorithms and Architectures* (Pittsburgh, PA, June 25–27). ACM Press, New York, 2012, 68–70.
35. Siek, J., Lee, L.-Q., and Lumsdaine, A. *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, Reading, MA, 2001.
36. Ugander, J., Karrer, B., Backstrom, L., and Marlow, C. *The Anatomy of the Facebook Social Graph*. Technical Report, 2011; <http://arxiv.org/abs/1111.4503>
37. Valiant, L.G. A bridging model for parallel computation. *Commun. ACM* 33, 8 (Aug. 1990), 103–111.
38. Xin, R.S., Gonzalez, J.E., Franklin, M.J., and Stoica, I. GraphX: A resilient distributed graph system on Spark. In *Proceedings of the First International Workshop on Graph Data Management Experiences and Systems* (New York, June 22–27). ACM Press, New York, 2013.

Andrew Lenharth (lenharth@ices.utexas.edu) is a research associate in the Institute for Computational Engineering and Sciences and a lecturer in the Department of Computer Science at the University of Texas at Austin.

Donald Nguyen (ddn@cs.utexas.edu) is a senior developer in Synthace Ltd., London, U.K.; he received his Ph.D. from the University of Texas at Austin in 2015 while doing research for this article.

Keshav Pingali (pingali@cs.utexas.edu) is the W.A. "Tex" Moncrief Chair of Grid and Distributed Computing, a professor in the Department of Computer Science, and a professor in the Institute for Computational Engineering and Sciences at the University of Texas at Austin.

© 2016 ACM 0001-0782/16/05 \$15.00



Watch the authors discuss their work in this exclusive Communications video. <http://cacm.acm.org/videos/parallel-graph-analytics>

DOI:10.1145/2904378

Comparing smartphone mapping apps leads to unexpected surprises.

BY HANAN SAMET, SARANA NUTANONG, AND BRENDAN C. FRUIN

Static Presentation Consistency Issues in Smartphone Mapping Apps

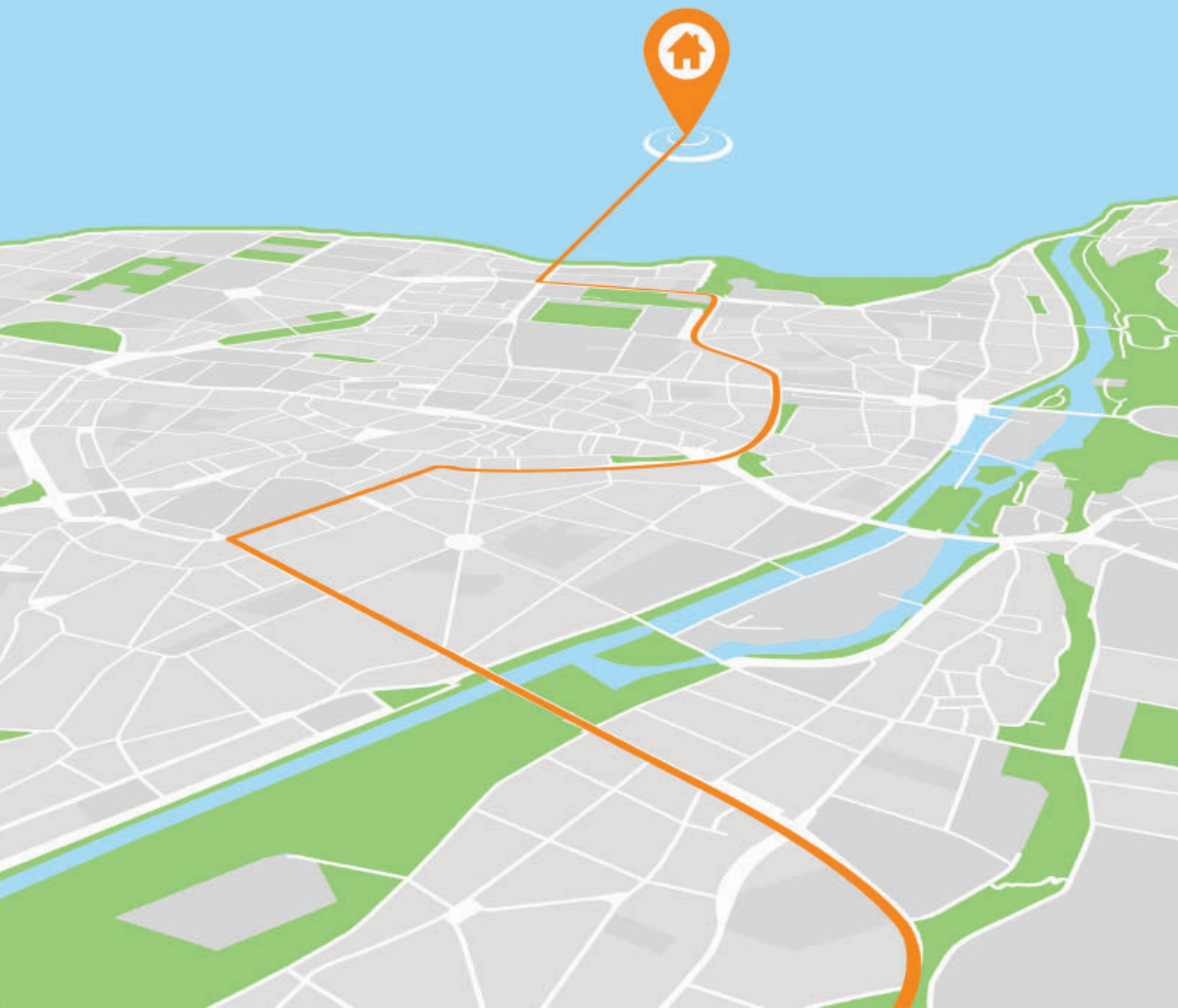
THE EXPLOSIVE GROWTH of the Internet coupled with the increasing use of location-enabled devices such as smartphones has led to an increasing awareness of the importance of location information, which traditionally has been presented with a map. For centuries, maps have been used to convey abstractions of spatial information in a manner that is aesthetically pleasing and familiar to their users. Often this came at the expense of accuracy, which users have found to be acceptable, usually due to conformance with

commonly held beliefs. For example, labels for place names are supposed to be placed on the map so they do not overlap names of other nearby places, and winding roads with switchbacks are represented with a screw-like symbol where the number of turns in the symbol usually has no correlation with the number of switchbacks actually present. In the past, maps were used not only to present information but also to store information, and to provide easy and rapid access to it (also known as indexing using today's parlance.³²).

Traditionally maps were drawn by cartographers, who were often regarded as artists. This took a considerable amount of skill, effort, and time, and the maps are still highly valued from both financial and artistic perspectives. The advent of computers, and the increase in their use to produce maps, as well as the diversity and increasing sophistication of the output devices on which the maps are presented and viewed, led to a dramatic decrease in the time needed to produce maps, and hence in their variety and distribution. In particular, maps are no longer created and produced only when there was a sufficient demand for them, where "sufficient" was usually defined quantitatively. Moreover, maps are no longer necessarily printed nor assembled in collections such as atlases, often with a common theme, such as the display of particular attributes like crops, land use, rainfall, and so on. Instead, maps are produced in a custom-made manner to display some

» key insights

- Static presentation consistency properties motivated by centuries-old classical principles used by cartographers (label choice and placement) are applied to evaluate smartphone mapping apps.
- Smartphone mapping apps must take into account the small form factor that limits the screen real estate.
- Newer is not always better as is demonstrated for smartphone mapping apps.



specific spatial relationship rather than in groups, most often in units of one, and in a way that allows them to be manipulated.

The rise of the Web and the ease with which documents can be accessed, regardless of their physical location, has profoundly impacted the accessibility of maps and their customized generation and use. People do not hesitate to decide they need a map, and, in fact, results returned by search engines are often accompanied by a map when the result involves some location information. The re-

sults can be viewed dynamically, unlike atlases that are usually viewed statically, meaning that changes can be made through manipulation actions such as browsing (for example, Anselin,⁸ Esperança,¹² Samet³³) including panning and/or zooming, or manipulating what is termed a *spatial spreadsheet*.¹⁸ Moreover, the Web has made it easier to find and retrieve data by location (that is, index it) regardless of whether the location is specified explicitly or, increasingly more importantly, implicitly by virtue of the physical location of the user.

Location Specification

The explicit specification of location has traditionally been geometric (for example, as latitude-longitude pairs of numbers). This is often cumbersome, as users do not think of a location in this way. They often do not know it in this way or have easy access to it. More importantly, they are not accustomed to communicating it to others in this way. Instead, they are used to specifying a location textually (including verbally). A textual specification has a number of advantages. First, ease of communication especially on smartphone

devices where a textual (also increasingly verbal via speech recognition, such as Siri on the Apple iOS platform) input capability is usually present. Second, text acts like a polymorphic type in the sense that one size fits all. In particular, depending on the application, which makes use of this information, a term such as “Washington” can be interpreted both as a point or as an area, and the user need not be concerned with this question. The drawback of textual specification of location data is ambiguity. For example, there are many locations named “Washington,” and they must be resolved (known as toponym resolution^{7,22}). Moreover, in some cases we are not even sure the term “Washington” denotes a location as it could be a reference to the name of a person (known as toponym recognition^{21,30}). This can be the case when processing documents such as newspaper articles, tweets, blogs, and so on. The drawback can be overcome by taking advantage of the fact that toponyms often appear together as in lists or tables where we can make use of clues such as prominence, proximity, and sibling (for example, Adelfio,⁶ Lieberman²⁵). The process of understanding and converting a textual specification of a location to its geometric specification is known as geotagging (for example, Hoffart¹⁷) and is beyond the scope of this article.

Implicit specification of location can be done in a number of ways including by the IP address of the user’s computing platform (regardless of its size) or, increasingly by an embedded GPS capability that provides the user’s physical location.

Another technique of location specification that is increasingly used with the rising popularity of touch interfaces combines implicit and explicit specifications to yield an approximate specification. Observe that a map, coupled with the ability to pan and to vary the zoom level at which the world is viewed, provides an inherent granularity to the location specification process, which facilitates this approximate specification. In particular, the act of pointing at a location (that is, by the appropriate positioning of a pointing device with the aid of panning) and making the interpretation of the precision of this positioning



The Web has made it easier to find and retrieve data by location regardless of whether the location is specified explicitly or, increasingly more importantly, implicitly by virtue of the physical location of the user.



specification dependent on the zoom level is equivalent to permitting the use of spatial synonyms,^{23,38,41} which are the hallmarks of approximate specifications. For example, a user posing a query seeking a concert in Manhattan would be satisfied by a concert in Harlem by virtue of proximity, New York City by virtue of containment, and Brooklyn by being a sibling borough of Manhattan in New York. Thus, users no longer need to know the exact name or position of the sought location. In other words, the touch interface serves as an implicit access structure to the data accomplished with direct manipulation. Of course, an index must be built (for example, Hjaltason¹⁶) whose access is achieved by software that translates the screen coordinates (using nearest neighbor techniques as in a “pick” operation in computer graphics) to the ones used by the index.

Mapping Apps on Smartphones

The almost universal adoption of smartphones, and, to a lesser but increasing extent, tablet devices (virtually all of which have an embedded GPS) has made location information a cornerstone of queries. This has led to the reinforcement of the realization that the map is the most convenient way (especially on the smartphone with its limited display size) of presenting query results to users and also to formulate and specify the query. This leads to a wide range of applications and the use of a wide range of sources for the maps.

The drawbacks of this wide range of sources for the maps is that the maps generated by them are not always produced in a manner consistent with the traditional concerns for the factors of/trade-offs between accuracy, aesthetics, and completeness, as well as not being in line with generally accepted cartographic principles (for example, Robinson³¹). To a large extent, the airing (as well as increasingly venting) of these drawbacks has lain dormant in the sense that people were subconsciously aware of them but were so satisfied with the resulting increase in capabilities they were inhibited from expressing their disappointment in their failures to live up to them.

However, all of these inhibitions were abandoned with the introduction

of the Apple iPhone 5 smartphone and the accompanying iOS6 software environment. It replaced the use of a mapping app on Apple's mobile devices based on Google's map data (referred to here as the iOS5 mapping app) with an app that makes use of Apple's map data (referred to here as the iOS6 mapping app as well as the subsequently released iOS7 mapping app and iOS8 mapping app). It also changed decisions as to what data is displayed (served to the user) in responses to queries (especially implicit ones through the manipulation of the viewing window). This replacement has led to significant changes in the user experience with apps that both make use of and serve map data, and has resulted in closer scrutiny of mapping applications on mobile devices as done here.

The mapping applications on the mobile devices (smartphones and tablets) are not the traditional ones where the map is used in a passive manner as is the case in atlases containing maps that are browsed leisurely. On mobile devices, the map is used in an active manner as a tool to enable such tasks as navigation and location finding, using pan and zoom. Here accuracy is paramount, and now issues of data quality and lack of quality assurance policies and protocols by Apple in releasing the iOS6 mapping app became very apparent. This resulted in errors such as misplacing the town of Uckfield in East Sussex in the U.K.² as well as others.^{4,11} In fact, the public uproar over them was so large it led to the eventual dismissal of Apple's leaders of the new mapping app project.³ Most of these errors have been fixed in subsequent releases of iOS6 and in its iOS7 and iOS8 successors. Nevertheless, some persist, such as marking the city of Faro in Portugal as a park⁴ (see Figure 1a from iOS8 version 8.1.2).

Notwithstanding these resolved issues, at times, we have found the iOS6, iOS7, and iOS8 mapping apps to be lacking from the perspective of presentation consistency when deployed on mobile devices such as smartphones due to the limited amount of screen "real estate." For example, consider the map of Europe in an early version of the iOS6 mapping app given in Figure 1b. Here, the labeled countries are poorly distributed with a high concentration

in Eastern Europe while not labeling major countries such as Italy and France, although their capital cities are labeled. In addition, Algeria is mislabeled as "Alger." Surprisingly, criticism from such a perspective has rarely been leveled before (but see Paolino,²⁸ Samet³⁶), which we do in this study in greater detail using examples of how they also plague other mapping apps.

Motivation

The motivation for our study is to take advantage of the fact a map provides an efficient way of accessing spatially referenced data when we cannot look at all of it at once. Our observations are based on the experience we gained in building the STEWARD,²⁴ NewsStand,^{23,38,41} TwitterStand,^{15,19,40} PhotoStand,³⁷ and TweetPhoto¹⁴ systems and adapting them (especially NewsStand and TwitterStand) to run on smartphones.^{34,35} These systems access documents such as, but not limited to, news and photos with a map query interface (that is, by location and, to a lesser extent, also by topic). In these applications, as well as in many related ones, the map on the smartphone helps

users to anchor and orient answers to queries in which they want to take advantage of spatial synonyms. In addition, we are motivated by the desire to be able to place spatially referenced information on the map such as icons for topics, image thumbnails (for example, Nutanong,^{26,27} Peng²⁹), names of particular locations, names of people and diseases,²⁰ mentions of brands,⁵ or any other data that lends itself to being classified using an ontology.

Note the Gazetteer, which is used to translate textual specifications to geometric ones, can also be considered an ontology. The result is analogous to a mashup except, in our case, the mashup is hierarchical in the sense that as we zoom in on the map, additional spatially referenced information is displayed that was not of sufficient importance to be displayed when we zoom out completely. This zoom out capability is not available in comparable systems such as HealthMap^{9,13} for disease monitoring.

This article compares the mapping apps using various presentation consistency properties broken down into two categories: static and dynamic. We

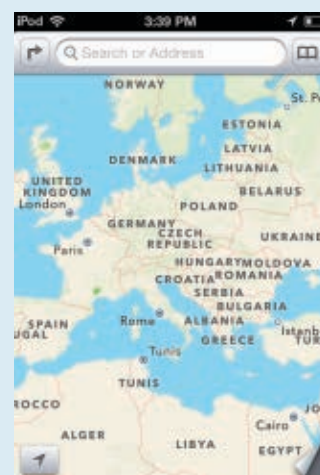
Figure 1. Different views.

(a) Faro, Portugal is represented as a park in iOS8 on an iPhone 5 (and also in iOS6 on an iPhone 5 and in iOS7 on an iPhone 4), while (b) shows a map of Europe in an early version of iOS6 on an iPad Touch with a high concentration of Eastern European countries while not labeling major countries such as Italy and France.



(a) i

iOS8 and iOS6 on iPhone 5 and iOS7 on iPhone 4



(b)

iOS6 Old on iPad Touch

focus on the static consistency properties although we also give a brief definition of the dynamic properties with more details provided elsewhere.³⁹ We also include a detailed description of the environments in which the comparison was conducted.

Comparison Summary

Our choices of presentation consistency properties are motivated by centuries-old classical principles used by cartographers that are derived from the static way in which maps have been traditionally browsed, as well as motivated by the continuously evolving dynamic ways maps are browsed that involve manipulation actions and have much to do with the platform used to view them (for example, an atlas versus a smartphone). The classical static presentation consistency issues involve the undesirability of label overlap and having a reasonable label distribution that are aesthetic in nature. Other static consistency properties include hierarchical consistency that simply seeks a consistent way of presenting labels of containing entities by requiring they must be included whenever they are visible in their entirety, while sibling consistency corresponds to labeling all visible spatial entities that are in the same level of the mapping object hierarchy. These properties pertain to issues related to the identity of the labels that should be present and the manner in which they are presented. They are the subjects of this article.

The dynamic presentation consistency properties are a result of the manipulation actions users take to browse the map and are often achieved by gesturing. They include pan, zoom, full

zoom out, and wraparound. The pan and zoom consistency properties correspond to the integrity of the available gesturing actions in terms of retaining label information as the actions take place. In particular, the premise is if a spatial entity has been labeled, then the label persists as long as the spatial entity remains visible in its entirety. The full zoom out property reflects the desire to be able to view the Earth in its entirety rather than being compelled to apply pan operations to do so. Similarly, the wraparound property is an acknowledgment the Earth is round and again reflects the desire not to have to apply pan operations in the opposite direction to be able to view adjacent locations on the map. These properties are discussed in greater detail elsewhere.³⁹

In terms of devices, we compare the iOS6 mapping app (initially iOS version 6.1 on iPhone 5 and most recently 6.1.4), the iOS7 mapping app (iOS version 7.0 on iPhone 4), the iOS8 mapping app (iOS version 8.1.2 on iPhone 5), the iOS5 mapping app (iOS version 5.1.1 on iPod Touch), the Android mapping app (Maps version 8.0.0 on Android 4.3), Google's iOS mapping app (version 1.0) for Google Maps data (referred here as the iOS mapping app for Google), and the HERE Maps app on Microsoft's Windows Phone 8 (HERE Maps version 3.5.481.8 with map data 8.0.50.116). Although each vendor's mapping apps are similar, they do not always yield the same result. At times, we also use the qualifiers "old" and "new" to distinguish between the versions of iOS6 used in our initial tests (version 6.1 in October 2012) and in our most recent tests (version 6.1.4 in April 2014 and later), respectively.

This distinction is necessary because we observed the algorithms used to implement the various mapping apps are frequently changed for a particular version of the operating system, even if the operating system is not updated. It is especially true for the label distribution and placement algorithms that we often found to yield different results for the same queries. This is because all queries are transmitted to the map tile server over the Internet, and the server makes the final decision as to what labels will be placed, and where, on the resulting map tiles that are also transmitted by the server. Therefore, do not be surprised if you cannot always repeat our observations. The important take away from them is the undesirable behaviors of some mapping apps should not be taken as absolutes but instead are just indications of what could possibly go wrong. Our comparison also contains the iOS Apps of Bing Maps, Nokia Maps, ESRI, MapQuest, and OpenStreetMap.^a

The accompanying table summarizes the comparison in terms of presentation consistency properties we want satisfied. Their satisfaction is primarily an issue on smartphones where the screen size is small thereby requiring panning and zooming for more information, while not needed on tablets where the screens are larger. The apps are detailed in this article and identified using I5 for iOS5, I6 for iOS6, I7 for iOS7, I8 for iOS8, A for Android, WP for HERE Maps on Windows Phone 8, IG for iOS of Google, IB for iOS of Bing Maps, IN for iOS of Nokia Maps, IQ for iOS of MapQuest, IO for iOS of OpenStreetMap, and IE for iOS of ESRI. The table denotes whether the property does not (×), partially (P), or holds (✓) for the apps.

Note the variation in the relative sizes of the screenshots in some of the

Consistency property comparison of mobile mapping apps.

Properties	Mapping Applications (Apps)											
	I6	I7	I8	I5	A	IG	IB	IN	WP	IQ	IO	IE
Hierarchical C	×	×	×	<i>P</i>	<i>P</i>	<i>P</i>	✓	<i>P</i>	×	×	✓	×
Sibling C	×	×	×	✓	×	×	×	×	×	×	✓	×
No Overlaps	✓	✓	✓	✓	✓	✓	<i>P</i>	✓	✓	✓	✓	×
Label Distribution	×	×	<i>P</i>	✓	✓	✓	×	×	×	✓	✓	✓
Panning C	×	×	×	✓	✓	✓	✓	✓	×	✓	✓	✓
Zoom C	×	×	×	✓	×	×	✓	✓	×	✓	✓	<i>P</i>
H Wraparound	✓	✓	✓	×	✓	✓	×	✓	✓	✓	✓	✓
V Wraparound	×	×	×	×	×	×	×	✓	×	✓	×	×
Full Zoom Out	<i>P</i>	<i>P</i>	<i>P</i>	✓	×	<i>P</i>	✓	×	✓	×	✓	✓

a In particular, we include the iOS apps of Bing Maps (version 3.03), Nokia Maps (HERE Maps version 1.8), which is also increasingly serving as the source for Bing Maps,¹ ESRI (ArcGIS version 2.3.2), MapQuest (version 3.3.1), and OpenStreetMap denoted by OSM (whose open source map data forms the basis of OpenSeaMap version 1.1, which is used here). We point out OpenStreetMap could have also been used as the source map data for the MapQuest app. Note the iOS mapping apps of Google, MapQuest, Nokia Maps, OSM, Bing Maps, and ESRI were all tested on iOS version 6.1.

figures due to the different devices we used. In particular, the screenshots for the Android and Windows Phone mapping apps are larger due to 2.5×4.3 and 2.31×3.86 inch screens for the Android and Windows Phone, respectively, instead of a 2×3 inch screen for the iPhone 4 and iPod Touch and a 2×3.5 inch screen for the iPhone 5 devices we used. The latter was used to perform comparisons with the iOS5 mapping app which is not available on the iPhone 5 as well as with the iOS mapping apps.

Hierarchical Consistency

Generally speaking, the name of a location should not be displayed without also displaying the name of its containing location provided the area spanned by the containing location is visible in its entirety (termed hierarchical consistency). Therefore, if Los Angeles is displayed, then so should the name of its containing state, California. While being desirable, examples can be found where this property does not hold for the iOS6, iOS7, iOS8, and Android mapping apps (see the discussion and examples of zoom consistency in Samet³⁹) as well as for the Windows Phone mapping app (for example, a scenario not shown here where Belgrade is displayed but not the name of its containing country, Serbia). In the case of the iOS mapping apps, it only holds uniformly for the Bing and OSM variants. For the iOS5 and Android mapping apps and for the iOS mapping apps for Nokia and Google, it holds only for Australia, Brazil, Canada, and the U.S., but does not hold for China, India, and Mexico. It completely fails to hold for the iOS mapping apps for MapQuest and ESRI.

Sibling Consistency

If the name of an object at a particular depth of the mapping hierarchy is displayed, then the names of all of its visible sibling objects should also be displayed and should use the same font type and point size or symbol (termed sibling consistency). In other words, if, for example, the name of one state is displayed, then the names of all visible states should be displayed using a consistent labeling scheme (for example, full name or abbreviation,

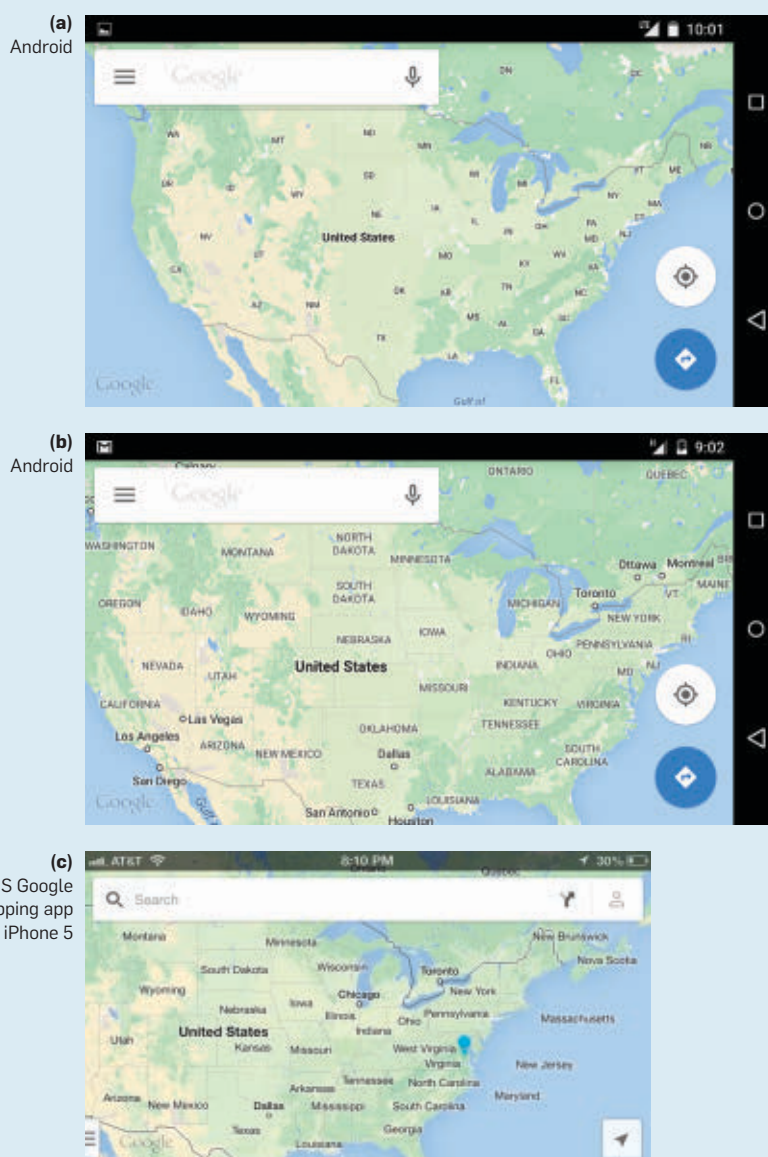
abbreviation type or style, all caps, boldface font, font point size) while also obeying a stipulation that the labels not overlap and that the name of the containing country be displayed as well (hierarchical consistency). We discuss the satisfaction of this property only for states and provinces within a country and for continents within a maximum zoom out level map of the world. We do not discuss it for other objects such as countries within continents for which the sibling consistency requirement is generally waived due to impracticality. We examine it

primarily for the U.S. and Canada. In our examples of countries, we often restrict ourselves to landscape maps as they maximize the amount of information that can be presented on the smartphone form factor.

For the U.S. map, none of the most popular mapping apps that are currently used satisfy the sibling consistency property in its entirety. For example, the Android mapping app (Figure 2a) and the almost equivalent iOS mapping app for Google (not shown here), at the level of zoom that permits the entire U.S. to be seen, use

Figure 2. U.S. maps using Google's mapping apps showing the absence of sibling consistency.

(a) Map from Android mapping app on Lollipop where all states are labeled using abbreviations. (b) Result of zooming in slightly on the map in (a) so most states are labeled in full although some are missing while for others only abbreviations are present. (c) Map from the iOS mapping app of Google on an iPhone 5 where labels of some states are erroneously placed in the Atlantic Ocean.



abbreviations and yield the same result. However, the sibling property is not satisfied for both of them as we see the names of two states (Rhode Island and Vermont) are missing.

Zooming in a bit further on the Android map in Figure 2a so the map spans almost the entire U.S. and applying a small amount of panning to see the remaining parts of the U.S. (see Figure 2b), finds that most of the names of the states are spelled out in full although some are abbreviated (Maryland, New Jersey, Rhode Island, and Vermont), while others are missing (Arkansas, Colorado, Connecticut, Delaware, Florida, Georgia, Illinois, Kansas, Massachusetts, Mississippi,

New Hampshire, North Carolina, West Virginia, and Wisconsin). In the case of Colorado and Kansas, one possible explanation for omitting them is the presence of the label corresponding to the name of the containing object (which is the United States) thereby satisfying hierarchical consistency; however, they could have been included as there is room for them. The hierarchical consistency property is also satisfied for the various cities present in the sense the names of their containing states are also included. The names of a number of some of the states that are missing such as Arkansas, Georgia, Illinois, North Carolina, and Wisconsin could have been includ-

ed as there is room for them.

The labels of some of the missing states in the Android mapping app (for example, Delaware, Massachusetts, and North Carolina), and of the abbreviated ones (for example, Maryland, New Jersey, and Rhode Island) could have fit by extending some parts of the labels to the adjoining body of water even without abbreviations as done, for example, for San Diego (see Figure 2b).^b

For the U.S. map, the iOS6, iOS7, and iOS8 mapping apps only display the names of a few cities as well as the name of the containing country (with the exception of iOS7), which is the U.S., but do not display the names of the containing states (see Figure 3a). Zooming in a bit further so the map spans almost the entire U.S. and applying a small amount of panning to see the remaining parts of the U.S. (see Figure 3b), finds the names of most of the states are labeled using a multitude of formats ranging from being fully spelled (Ohio, Iowa, Utah, Maine, Idaho, Texas), partially spelled out (for example, Tenn for Tennessee), specified using standard abbreviations (for example, KY for Kentucky), and abbreviated with periods (for example, N.C. for North Carolina). Notice the name of the containing country is present and does not prevent the labeling of any states in its vicinity although some visible states are missing (Connecticut, Delaware, Maryland, Mississippi, New Jersey, and West Virginia). Clearly the sibling consistency property is not satisfied, although the hierarchical consistency property is satisfied.

Contrast the iOS8 mapping app with the iOS5 mapping app (see Figure 3c), where all states are labeled

^b Special care must be exercised in doing this as shown in Figure 2c where Maryland, New Jersey, and Massachusetts are located in the Atlantic. This was encountered in May 2014 in the Android mapping app running on Android Google Maps version 8.0.0, as well as the Google mapping app for iOS, although it could not be repeated as of July 25, 2014 when running on Android Google Maps version 8.2.0 and also not in the Google mapping app for iOS. Interestingly, the same version of the Android mapping app was used on both dates (3.0.1.23805). One possible reason for the discrepancy is including improvements in the map labeling algorithms deployed by the server even though new versions of the Android and iOS Google mapping apps had yet to be released.

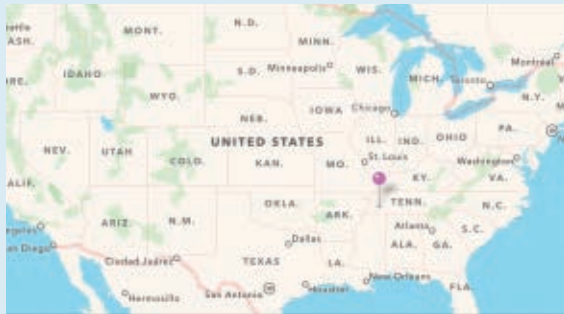
Figure 3. U.S. maps using Apple mapping apps.

(a) iOS8 mapping app on an iPhone 5 where only cities are labeled. (b) Result of zooming in slightly on the map in (a) where states are labeled using a combination of spelled out names and abbreviations. (c) iOS5 mapping app on an iPod Touch where all states are labeled using USPS abbreviations and hence satisfy sibling consistency.

(a)
iOS8 on iPhone 5



(b)
iOS8 on iPhone 5



(c)
iOS5 on iPod Touch



using the USPS (U.S. Postal Service) abbreviations placing them in the ocean and adding appropriate pointers when there is not enough room. Notice the containing country is included but no cities are labeled. Both sibling and hierarchical consistency are satisfied by the iOS5 mapping app.

Of the remaining iOS mapping apps (not shown here), only OSM satisfied these two consistency properties as most of the data requires a significant amount of zooming in to see and the non-overlap property is not satisfied. The iOS mapping app for Bing labels the states using both fully spelled out names and abbreviations all with a standard font, but does not use the same font point size. Its novelty lies in the use of a watermark-style font for the containing country thereby enabling it to overlap the names of the states although a number of states are still not labeled. The iOS mapping app for Nokia makes use of standard abbreviations but misses a number of states and Canadian provinces due to labeling of the container U.S. with “United States of America,” as well as the containing continent “North America,” both in a relatively large point size watermark style. Unfortunately, the states and provinces are also labeled using a watermark-style font and thus the nonoverlapping label requirement prevents some of them from being labeled unlike the iOS mapping app for Bing. The iOS mapping apps for MapQuest and ESRI make no attempt to present a map of the states of the U.S.

Presently, the Windows Phone mapping app cannot be used in landscape mode and thus we only review its operation for the U.S. map in portrait mode where it uses abbreviations for the names of the states and labels some of the cities in which case it also labels the containing states. In addition, it labels the containing country although this is at the expense of omitting the names of some of the states (Iowa and Nebraska) as in the case of the Android mapping app even though there is room for them.

Overlap Avoidance

A classical property is that labels of place names should not overlap (for example, Christensen¹⁰). It is enforced in all of the mapping apps with the exception of the iOS mapping app of OSM

that permits labels to overlap and the iOS mapping app of Bing where watermark-style labels are permitted to overlap other labels with different fonts. For example, the label “Africa” in the map of Africa in the iOS mapping app of Bing in Figure 4a is a watermark and is overlapped by Cameroon, Gabon, and Congo in a standard font.

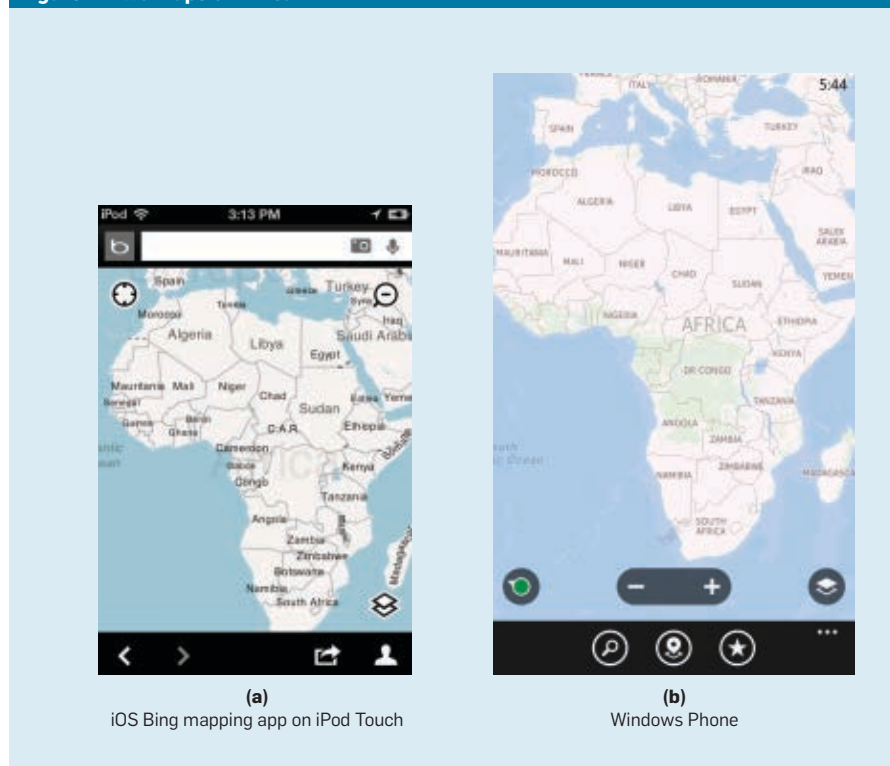
Note the iOS6, iOS7, and iOS8 mapping apps also make use of watermark-style labels for very large regions vis-a-vis the size of the display screen such as neighborhoods, cities, and countries (but, surprisingly, not states) apparently in order to provide contrast. However, they do not appear to permit watermark-style labels be overlapped by labels corresponding to names of objects of other types such as roads, neighborhoods, and cities, respectively, that are labeled using other font styles such as standard or boldface (not shown here). Interestingly, for example, continents are labeled using a watermark style in the iOS6 mapping app (not shown here) but not in the subsequently released iOS7 and iOS8 mapping apps where continents are labeled using a boldface font all caps with widely spaced letters (see Figures 3a and 3b). All but the iOS mapping app of Bing appear to restrict the orientation of the labels to be horizontal and parallel to

the x coordinate axis. In particular, the iOS mapping app of Bing uses curved labels for Madagascar, Malawi, and Somalia (Figure 4a). It is also noteworthy in using different font point sizes for the labels (compare Algeria with Gabon) as well as abbreviations (C.A.R., for the Central African Republic).

Curiously, the iOS mapping app for Bing (Figure 4a) uses a watermark-style label for Africa yet it does not label the Democratic Republic of the Congo. This is so even though it could have done so easily in the same manner as done in the Windows Phone mapping app (see Figure 4b) and the iOS mapping app for Nokia (not shown here as it is identical to the Windows Phone mapping app in Figure 4b). On the other hand, these two mapping apps do not label C.A.R. as it would overlap Africa, which is not allowed as they use the same watermark-style font to label all objects instead of restricting it to continent names as in the iOS mapping app for Bing.

Interestingly, although Nokia's HERE Maps is increasingly serving as the source for Bing Maps¹ and is now used in the Windows Phone mapping app, the quality and extent of the information displayed by the Windows Phone mapping app has declined vis-a-vis what was previously avail-

Figure 4. Two maps of Africa.



able on the iOS mapping app for Bing. For example, as we pointed out previously, although the iOS mapping mpp for Nokia makes use of a watermark-style label, it deploys it for all of the objects and this is what is done in the Windows Phone mapping app and the iOS mapping app for Nokia (see Figure 4b). Thus, they abandon the increased functionality afforded by the watermark-style font, which can be seen by the fact that fewer countries are labeled in the Windows Phone mapping app than in the iOS mapping app for Bing.

Label Distribution

Besides not being encouraged to overlap, labels should also be well distributed rather than being bunched up in one or a few regions of the map while the rest of the map contains just a few, if any, labels. The maps of Europe in Figure 1b and Figure 5 demonstrate some stark differences between the iOS6, iOS7, iOS8, iOS5, and Windows Phone mapping apps with respect to label distribution. We do not show the Android and Google mapping apps for iOS as they do not differ from the iOS5 mapping app for this example. For the original and early releases of the iOS6 mapping app, some of the issues we mention appear to be fixed in subsequent releases of its successor iOS6, iOS7, and iOS8 mapping apps and thus we differentiate between the variants of iOS6 using the qualifiers “old” (November 2012) and “new” (January 2015).

We immediately notice the two iOS6 (Figures 1b and 5a) and the iOS7 (Figure 5b) mapping apps do a poor job of label distribution. Moreover, countries that are labeled are not necessarily the most important or the most populated. For example, France and Italy are absent in the old iOS6 mapping app (Figure 1b); Italy and the United Kingdom are absent in the new iOS6 mapping app (Figure 5a); Syria is absent in the iOS7 mapping app (Figure 5b). In particular, it appears the countries near the center of the viewing window are given preference. The iOS8 mapping app (Figure 5c), while showing a much better label distribution than its predecessor iOS6 and iOS7 mapping apps, is still lacking where we see the absence



We observed the algorithms used to implement the various mapping apps are frequently changed for a particular version of the operating system, even if the operating system is not updated.



of labels in Central Europe (Austria and/or Switzerland). Also, Serbia is absent while it can fit and it was present in the two variants of its predecessor iOS6 and iOS7 mapping apps.

The iOS5 mapping app (Figure 5d) does a better job of deciding that at this level of zoom in, the country entity is more relevant and the labeled countries are chosen on the basis of which ones provide the best distribution over the space spanned by the query window. This is better than the approach of the iOS6 and iOS7 (but not the iOS8) mapping apps of displaying names of countries without much thought paid to their spacing and also to their population, thereby leading to a cluttered appearance. A similar label distribution problem arises in the Windows Phone mapping app where only the cities Paris, Lyon, Marseilles, and Toulouse are displayed for France while many cities are displayed for Germany (Figure 5e).

Note the mixture of city and country names in the two iOS6 and the iOS7 (and to a lesser extent the iOS8) mapping apps without satisfying hierarchical consistency. In particular, when a city name appears, the name of the country is not always included such as Rome and not Italy in the two iOS6 mapping apps (Figures 1b and 5a), Prague and not the Czech Republic in the iOS7 mapping app (Figure 5b), and Zagreb and not Croatia in the iOS8 mapping app (Figure 5c). This also occurs in the Windows Phone mapping app where Belgrade is included but not Serbia (not shown here).

Concluding Remarks

We now review some of the presentation consistency issues for the various mapping apps we found noteworthy with an emphasis on the static consistency issues although we do include the dynamic consistency issues when speaking about the properties in a collective sense. However, we first re-emphasize our aim here is not to criticize Apple, Google, or Microsoft. Instead, it is to use examples motivated by Apple's foray into the Maps space, where Google and Microsoft have a longer history due in part to their work on Microsoft Virtual Earth and Google Earth and Maps, to point out the difficulty of such a task and the

need to take into account centuries-old lessons in map making.

Despite the obvious similarities among the Android and iOS5 mapping apps, we saw important differences including the way they differ for the landscape U.S. map where the iOS5 mapping app labels all states while the Android mapping app only labels those that fit without conflict thereby omitting labels for small

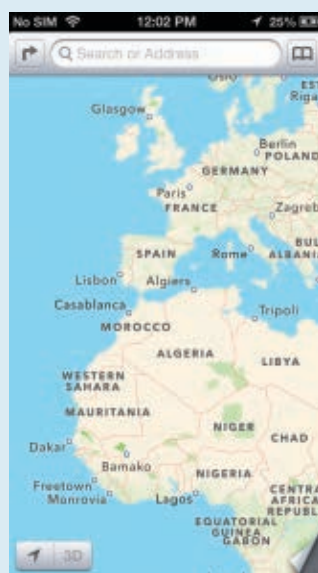
states and thus not satisfying sibling consistency.

From our limited comparison summarized in the table, we conclude that newer is not always better in the sense the iOS5 mapping app is probably still the best especially with respect to the four presentation consistency (hierarchical, sibling, panning, and zoom) properties making for better map-based applications

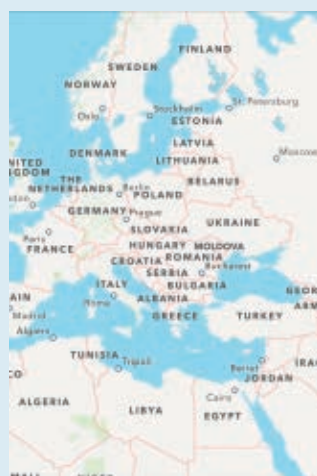
on smartphone form factor devices. The Android mapping app often has similar behavior although it was also plagued by zoom and sibling inconsistency that are also common to the iOS6, iOS7, iOS8, and Windows Phone mapping apps as well as others.

From an overall perspective, mapping apps from Google (iOS5 and Android) exhibit an understanding and appreciation of the small form fac-

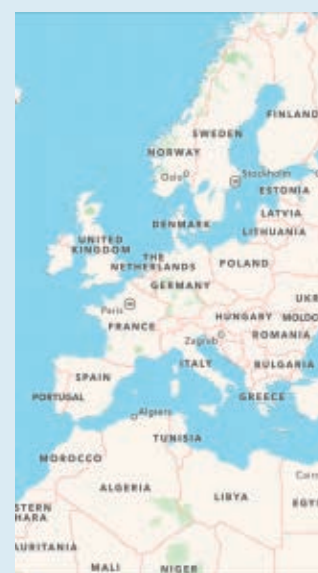
Figure 5. Example Europe maps showing the distribution or lack thereof for labels.



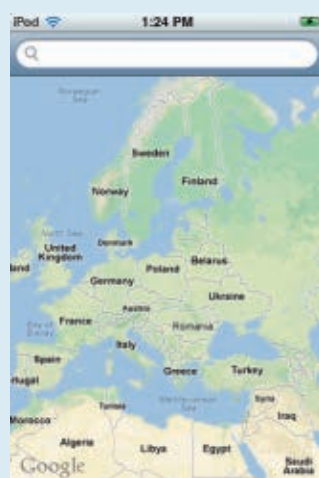
(a)
iOS6 New on iPhone 5



(b)
iOS7 on iPhone 4



(c)
iOS8 on iPhone 5



(d)
iOS5 on iPod Touch



(e)
Windows Phone

tor of the smartphone target device, which is easily seen by the nice distribution of labels of place names over the display screen. This is in contrast to the iOS6 and iOS7 (and to a far lesser extent in iOS8) mapping apps where, at times, the placement (and distribution) algorithm for the labels of place names is relatively poor. This is partly due to their use of large point sizes with fixed-width fonts and much space between the letters, although these factors are less of an issue on tablet devices (for example, iPad) that have a much larger form factor. Some of the remaining apps, at times, ignore the small form factor of the target device and the choice of the size and number of labels to display is made under the assumption the form factor of the target device is large (for example, a display monitor or even a tablet such as the iPad), or decide to display very few, if any, labels. Note that displaying too many labels reduces the app's utility to anchor additional spatially referenced information such as icons that is a critical requirement for mashups (for example, NewsStand^{34,35,41}) and is done so well on the mapping apps from Google (iOS5 and Android). Observe that use of watermark-style labels can mitigate the busy screen somewhat, but their use should not be all encompassing as in the Windows Phone app (Figure 5e).

It is important to note the main emphasis of this article has been to point out static consistency issues in well-known mapping apps. Ideally, any mapping app should satisfy all of the presentation consistency properties outlined in this article and tabulated in the table. In practice, however, an app developer may choose to partially satisfy or even completely ignore some of the properties due to factors such as space and computational time. In some cases, trade-offs must be made, and a possible future research direction is a study of how to make such a trade-off without compromising user experience. Such a study would require access to different labeling algorithms and involve usability testing to assess them.

Acknowledgments

This article is based on an earlier paper by Samet et al.³⁶ This work was sup-

ported in part by the National Science Foundation under Grants IIS-10-18475, IIS-12-19023, and IIS-13-20791. 

References

1. Nokia's Bing Maps deal is a sign of the future; http://rethink-wireless.com/print.asp?article_id=23446.
2. Apple's Maps app slammed over missing cities, other mistakes; http://www.cbsnews.com/8301-501465_162-57516870-501465/.
3. Bloomberg: Apple fires guy responsible for crappy Apple Maps; <http://gizmodo.com/5963651/apple-fires-guy-responsible-for-apple-maps>.
4. The Amazing iOS 6 Maps; <http://theamazingios6maps.tumblr.com/>.
5. Abdelrazek, A., Hand, E. and Samet, H. Brands in NewsStand: Spatiotemporal browsing of business news. In *Proceedings of GIS*, 2015, article 97.
6. Adelfio, M.D. and Samet, H. Structured toponym resolution using combined hierarchical place categories. In *Proceedings of the 2013 Workshop on Geographic Information Retrieval*, 49–56.
7. Amitay, E., Har'el, N., Sivan, R. and Soffer, A. Web-a-Where: Geotagging Web content. In *Proceedings of the 27th Annual International ACM SIGIR Conference on RGD in Information Retrieval*, 2004, 273–280.
8. Anselin, L., Kim, Y.W. and Syabri, I. Web-based analytical tools for the exploration of spatial data. *IJGIS* 6, 2 (2004), 197–218.
9. Children's Hospital Boston. Healthmap: Global Health, Local Knowledge. <http://www.healthmap.org/>, Sept. 2012.
10. Christensen, J. Marks, J. and Shieber, S.M. An empirical study of algorithms for point-feature label placement. *Trans. on Graphics* 14, 3 (1995), 203–232.
11. Dobson, M. Google maps announces a 400 year advantage over Apple maps; <http://blog.telemapics.com/?p=399>.
12. Esperança, C. and Samet, H. Experience with SAND-Tol: A scripting tool for spatial databases. *JVLIC* 13, 2 (2002), 229–255.
13. Freifeld, C.C., Mandl, K.D., Reis, B.Y. and Brownstein, J.S. Model formulation: Healthmap: Global infectious disease monitoring through automated classification and visualization of internet media reports. *JAMIA* 15, 2 (2008), 150–157.
14. Fruin, B.C., Samet, H. and Sankaranarayanan, J. Tweetphoto: Photos from news tweets. In *Proceedings of SIGSPATIAL*, 2012, 582–585.
15. Gramsky, N. and Samet, H. Seeder finder - identifying additional needles in the Twitter haystack. In *Proceedings of LBSN*, 2013, 44–53.
16. Hjalton, G.R. and Samet, H. Speeding up construction of PMR quadtree-based spatial indexes. *VLDJ* 11, 2 (2002), 109–137.
17. Hoffart, M.A. et al. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods for Natural Language Processing*, 782–792.
18. Iwerks, G.S. and Samet, H. The spatial spreadsheet. In *The Proceedings of VISUAL*, 1999, 317–324.
19. Jackoway, A., Samet, H. and Sankaranarayanan, J. Identification of live news events using Twitter. In *Proceedings of LBSN*, 2011, 25–32.
20. Lan, R., Lieberman, M.D. and Samet, H. The picture of health: Map-based, collaborative spatio-temporal disease tracking. In *Proceedings of HealthGIS*, 2012, 27–35.
21. Lieberman, M.D. and Samet, H. Multifaceted toponym recognition for streaming news. In *Proceedings of the Annual International ACM SIGIR Conference on RGD in Information Retrieval*, 2011, 843–852.
22. Lieberman, M.D. and Samet, H. Adaptive context features for toponym resolution in streaming news. In *Proceedings of the Annual International ACM SIGIR Conference on RGD in Information Retrieval*, 2012, 731–740.
23. Lieberman, M.D. and Samet, H. Supporting rapid processing and interactive map-based exploration of streaming news. In *Proceedings of SIGSPATIAL*, 2012, 179–188.
24. Lieberman, M.D. and Samet, H., Sankaranarayanan, J. and Sperling, J. STEWARD: Architecture of a spatio-textual search engine. In *Proceedings of GIS*, 2007, 186–193.
25. Lieberman, M.D. and Samet, H. and Sankaranarayanan, J. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *Proceedings of the 2010 Workshop on Geographic Information Retrieval*, article 6.
26. Nutanong, S., Adelfio, M.D. and Samet, H. Multi resolution select distinct queries on large geographic point sets. In *Proceedings of SIGSPATIAL*, 2012, 159–168.
27. Nutanong, S., Adelfio, M.D. and Samet, H. An efficient layout method for a large collection of geographic data entries. In *Proceedings of EDBT*, 2013, 717–720.
28. Paolino, L., Romano, M., Tortora, G. and Vitiello, G. Spatial data visualization on mobile interface - a usability study. In *Proceedings of IWCMC*, 2013, 959–963.
29. Peng, S.F., Adelfio, M.D. and H. Samet, H. Viewing streaming spatially referenced data at interactive rates. In *Proceedings of SIGSPATIAL*, 2014, 409–412.
30. Purves, R.S. et al. The design and implementation of SPIRIT: a spatially aware search engine for information retrieval on the Internet. *IJGIS* 21, 7 (2007), 717–745.
31. Robinson, A.H., Morrison, J.L., Muehrcke, P.C., Kimerling, J. and Guptill, S.C. *Elements of Cartography*. Wiley, New York, 6th edition, 1995.
32. Samet, H. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, CA, 2006.
33. Samet, H. et al. Use of the SAND spatial browser for digital government applications. *Commun. ACM* 46, 1 (Jan. 2003), 61–64.
34. Samet, H., Adelfio, M.D., Fruin, F.C., Lieberman, M.D. and Teitler, B.E. Porting a Web-based mapping application to a smartphone app. In *Proceedings of GIS*, 2011, 525–528.
35. Samet, H., Teitler, B.E., Adelfio, M.D. and Lieberman, M.D. Adapting a map query interface for a gesturing touch screen interface. In *Proceedings of the 2011 WWW Conference (Companion Vol.)*, 257–260.
36. Samet, H., Fruin, B.C. and Nutanong, S. Diving it out at the smartphone mobile app mapping API coral: Apple, Google, and the competition. In *Proceedings of MOBISIG*, 2012, 41–48.
37. Samet, H., Adelfio, M.D., Fruin, M.D., Lieberman, M.D. and Sankaranarayanan, J. PhotoStand: A map query interface for a database of news photos. *PVLDB* 6, 12 (2013), 1350–1353.
38. Samet, H. et al. Reading news with maps: The power of searching with spatial synonyms. *Commun. ACM* 57, 10 (Oct. 2014), 64–77.
39. Samet, H., Nutanong, S. and Fruin, B.C. Dynamic presentation consistency issues in smartphone mapping apps. *Commun. ACM*, to appear.
40. Sankaranarayanan, J., Samet, H., Teitler, B.E., Lieberman, M.D. and Sperling, J. Twitterstand: News in tweets. In *Proceedings of GIS*, 2009, 42–51.
41. Teitler, B.E., Lieberman, M.D., Panozzo, D., Sankaranarayanan, J., Samet, H. and Sperling, J. NewsStand: A new view on news. In *Proceedings of GIS*, 2008, 144–153.

Hanan Samet (hjs@cs.umd.edu) is a distinguished university professor in the Computer Science Department, Center for Automation Research, and Institute for Advanced Computer Studies at the University of Maryland, College Park, MD.

Sarana Nutanong (nutanong@umiacs.umd.edu) is an assistant professor in the Computer Science Department, at the City University of Hong Kong, Hong Kong SAR, China; his research for this article was conducted while he was a research associate at the University of Maryland, College Park, MD.

Brenda C. Fruin (bcfruin@gmail.com) is a software engineer at Zillow, Seattle, WA; his research for this article was conducted while he was a student at the University of Maryland, College Park, MD.



ACM Books



MORGAN & CLAYPOOL
PUBLISHERS

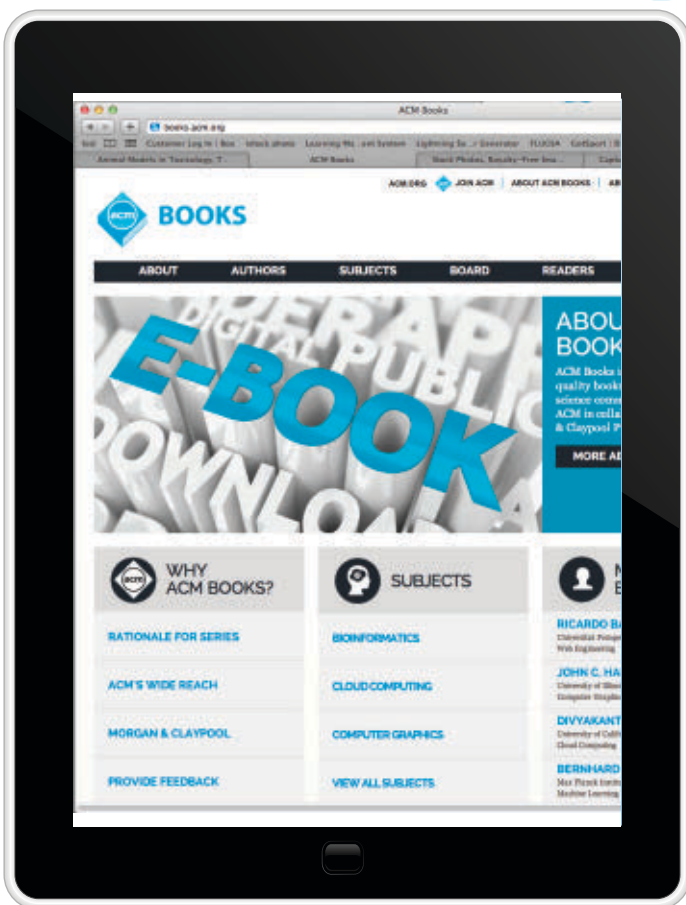
Publish your next book in the ACM Digital Library

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 Özsu), 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 Özsu, University of Waterloo, and a distinguished editorial board representing most areas of CS.

Proposals and inquiries welcome!

Contact: **M. Tamer Özsu**, Editor in Chief
booksubmissions@acm.org



Association for
Computing Machinery

Advancing Computing as a Science & Profession

Reviewing the technologies that enable robot musicians to jam.

BY MASON BRETAN AND GIL WEINBERG

A Survey of Robotic Musicianship

THE TERM ‘ROBOTIC MUSICIANSHIP’ may seem like an oxymoron. The first word often carries negative connotations in terms of artistic performance and can be used to describe a lack of expressivity and artistic sensitivity. The second word is used to describe varying levels of an individual’s ability to apply musical concepts in order to convey artistry and sensitivity beyond the facets of merely reading notes from a score.

To understand the meaning of robotic musicianship, it is important to detail the two primary research areas of which it constitutes: Musical mechatronics, which is the study and construction of physical systems that generate sound through mechanical means;¹⁵ and machine musicianship, which focuses on developing algorithms and cognitive models representative of various aspects of music perception, composition, performance, and theory.³¹ Robotic musicianship refers to the intersection of these areas. Researchers within this space design music-playing robots with the necessary underlying musical intelligence to support both expressive

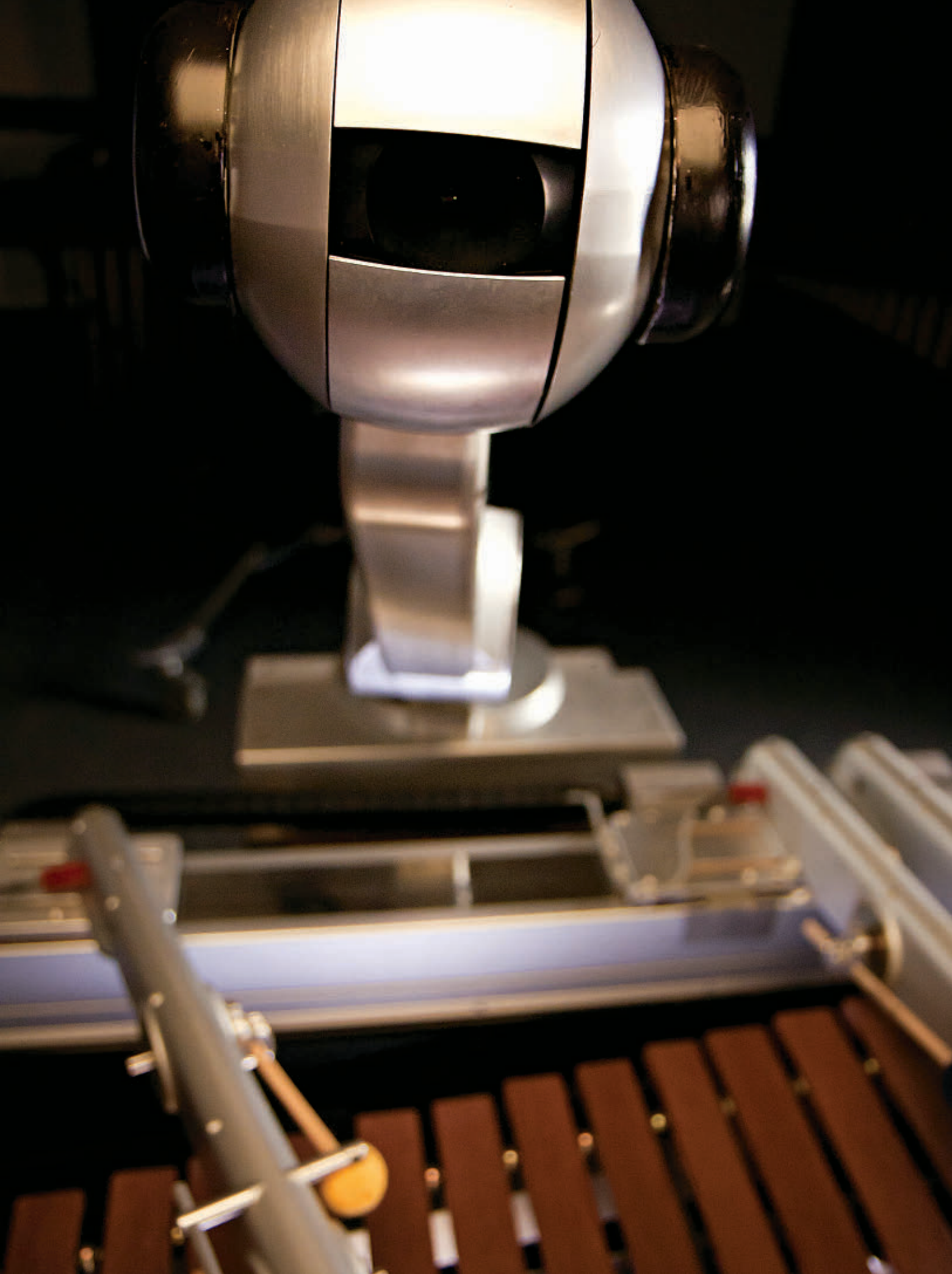
solo performance and interaction with human musicians.³⁷

Research in robotic musicianship has seen growing interest from both scientists and artists in recent years. This can be attributed to the domain’s interdisciplinary characteristics amenable to a wide-breadth of scientific and artistic endeavors. First, music provides an excellent platform for research because it allows us to transcend certain cultural barriers through common concepts such as rhythm and pitch. Second, robotic musicianship requires the integration of a wide variety of tasks, which combine aspects of engineering, computation, music, and psychology. Scholars in the field are drawn to this interdisciplinary nature and the inherent challenges involved in making a machine artistically expressive. Third, developing computer models for the perceptual and generative characteristics that support robotic musicianship can have broader impacts outside of artistic applications. Issues regarding timing, anticipation, expression, mechanical dexterity, and social interaction are pivotal to music and have numerous other functions in science as well.

Some researchers are drawn to robotic musicianship due to its potential for impact in other fields, ranging from cognitive science, human-robot

» key insights

- Robotic musicianship focuses on the construction of machines capable of producing sound, analyzing music, and generating music in such a way that allows them to showcase musicality and interact with human musicians.
- Research in robotic musicianship is motivated by the potential to enrich human musical culture and expand human creativity rather than replace it.
- Some robotic musicians are designed to play traditional instruments while others are designed to play augmented versions of instruments or even entirely new musical interfaces.
- Robotic musicians employ AI for identifying higher-level musical features essential to human musical cognition.



interaction (HRI), to human anatomy. Music is one of the most time-demanding mediums where events separated by mere milliseconds are noticeable by human listeners. Developing HRI algorithms for anticipation or synchronization in a musical context can, therefore, become useful for other HRI scenarios, where accurate timing is required. Moreover, by building and designing robotic musicians scholars can better understand the sophisticated interactions between the cognitive and physical processes in human music making. By reconstructing sophisticated sound production mechanisms, robotic musicians can also help shed light on the role of embodiment in music making—how music perception and cognition interact with human anatomy and physicality.

There are also cultural and artistic motivations for developing robotic musicians. Robert Rowe, a composer and pioneer in the development of interactive music systems, writes, “if computers interact with people in a musically meaningful way, that experience will bolster and extend the musicianship already fostered by traditional forms of music education...and expand human musical culture.”³¹ Likewise, the goal of robotic musicianship is not only to imitate human creativity or replace it, but also to supplement it and enrich the musical experience for humans. Exploring the possibilities of combining computers with physical sound generators to create systems capable of rich, acoustic sound production; intuitive, physics-based visual cues from sound producing movements; and, expressive physical behaviors through sound accompanying body movements is an attractive venture for many artists. Additionally, there is artistic potential in the non-human characteristics of machines including increased precision,

freedom of physical design, and the ability to perform fast computations. Discovering how best to leverage these characteristics to benefit music performance, composition, consumption, and education is a worthy investment as music is an essential component of human society and culture.

Another motivation for research in robotic musicianship stems from recent efforts to integrate personal robotics into everyday life. As social robots begin to penetrate the lives and homes of the public, music can play an important role in aiding the process. Music, in one form or another, is shared, understood, and appreciated by billions of people across many different cultures and epochs. Such ubiquity can be leveraged to yield a sense of familiarity between robots and people by using music to encourage human-robotic interaction. As more people begin to integrate robots into their everyday lives, trust and confidence in the new technology can be established.

But the recent interest in robotic musicianship has not come without popular criticism and misconceptions, mainly focused on the concern the ultimate goal of this research is to replace human musicality and creativity.^{a,b,c} Though much of this concern may stem from the natural awe and unease accompanied by the introduction of new technology, it is important to reiterate both our own and Rowe’s sentiment that by placing humans alongside artistically motivated machines we are establishing an environment in which human creativity can grow, thus, enriching human musical culture rather than replacing it.

A related criticism comes from the belief the creation of music—one of the most unique, expressive and creative, human activities—could never be replicated by automated “soul-less” machines, at least not “good music.”^d As veteran researchers in the field we be-

lieve robotic musicians bear great potential for enhancing human musical creativity and expression, rather than hindering it. For example, although in some cases the need for live musicians diminished when new technology was introduced (such as the introduction of talking films, or the MIDI protocol), technology has also introduced new opportunities in sound and music making, such as with sound design and recording arts. Similarly, disk jockeys introduced new methods for expressive musical performance using turntables and prerecorded sounds. Just as these new technologies expanded artistic creation, robotic musicianship has the potential to advance the art form of music by creating novel musical experiences that would encourage humans to create, perform, and think about music in novel ways. From utilizing compositional and improvisational algorithms that humans cannot process in timely manner (using constructs such as fractals or cellular automata) to exploring mechanical sound production capabilities that humans do not possess (from speed to timbre control), robotic musicians bear the promise of creating music that humans could never create by themselves and inspire humans to explore new and creative musical experiences, invent new genres, expand virtuosity, and bring musical expression and creativity to uncharted domains.

In this article we review past and ongoing research in robotic musicianship. We examine different methods for sound production and explore machine perception and the ability to decipher relevant musical events using various sensors. We discuss generative algorithms that enable the robot to complete different musical tasks such as improvisation or accompaniment. We then consider the design of social interactions and visual cues based on embodiment, anthropomorphism, and expressive gestures.

Sound Production and Design

For robots that play traditional acoustic instruments, typically the goal is to either emulate the human body and natural mechanics applied to playing an instrument or to augment an instrument to produce sound and create a physical affordance that is not possible in the instrument’s archetypal state,

Source Material

Readers are encouraged to check out the appendix accompanying this article in ACM’s Digital Library (see Source Material) for more examples, more technologies, and more information on the projects mentioned throughout this article as well as learn about future interfaces in robotic musicianship.

a <http://www.asylum.com/2011/01/19/robot-rock-bands-expressive-machines-musical-instruments/>

b <http://nymag.com/news/intelligencer/robot-jobs-2013-6/>

c <http://www.wired.com/2012/12/ff-robots-are-already-replacing-us/2/>

d <http://www.smithsonianmag.com/history/musicians-wage-war-against-evil-robots-92702721/?no-ist>

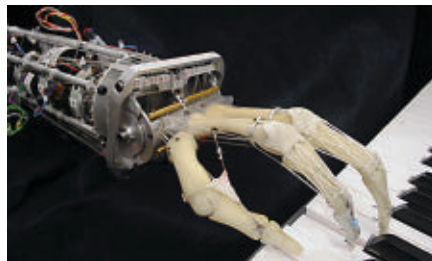
such as a prepared piano. (For a detailed review of robotic musical instruments see Kapur¹⁵).

Methods for playing traditional instruments. Traditional acoustic instruments are designed for humans. Robots are not constrained to human form and, thus, can be designed in a number of ways. Emulating people can be mechanically challenging, but sometimes it is helpful to have an established reference inspired by human physiology and the various physical playing techniques employed by musicians. However, new types of designs allow researchers to break free from the constraints of human form and explore aspects of instrument playability related to speed, localization, and dynamic variations not attainable by humans. In this section we will explore examples of different design methodology for different instrument types and discuss their inherent advantages and disadvantages.

Percussive instruments. The mechanics involved in human drumming are complex and require controlled movement of and coordination between the wrist, elbow, and fingers. Subtle changes in the tightness of one's grip on the drumstick, rate at which the wrist moves up and down, and distance between stick and drumhead give drummers expressive control over their performance. Human drummers use different types of strokes to achieve a variety of sounds and playing speeds. For example, for creating two hits separated by a large temporal interval, a drummer uses a single-stroke in which the wrist moves up and down for each strike. For two hits separated by a very small temporal interval a drummer may use a multiple-bounce stroke in which the wrist moves up and down once to generate the first strike, while subsequent strikes are achieved by adjusting the fingers' grip on the drumstick to generate a bounce. Multiple motors, gears, solenoids, and a sophisticated control system are required to imitate such functionality.

No systems exist that emulate natural multiple-bounce stroke drumming technique, however, several systems have been designed to imitate velocity curves of single-stroke wrist and arm movements. The Cog robot from MIT uses oscillators for smoother rhythmic arm control.⁴⁰ Another approach

to smooth motion control is to use hydraulics as done by Mitsuo Kawato in his humanoid drummer.² Sometimes robot designs are inspired more by the general physical shape of humans rather than specific human muscle and joint control. The humanoid keyboard playing WABOT has two hands for pressing keys and feet for pushing pedals.¹⁹ More recently Yoky Matsuoka developed an anatomically correct piano-playing hand.⁴¹



Piano-playing hand from the University of Washington.

Many designers of robotic percussion instruments can achieve their sonic and expressive goals using a much simpler engineering method than what is needed in order to replicate human physical control. A common approach for robots (not constrained to human form) that play percussive instruments is to use a solenoid system that strikes percussion using a stick or other actuators. Erik Singer's LEMUR bots³³ and portions of Orchestrion (designed for Pat Metheny⁶) use such an approach.

Singer's projects as well as others use non-anthropomorphic designs in which individual solenoids are used to strike individual drums. In addition to a simplified actuation design, non-humanoid robots can produce musical results that are not humanly possible. For example, in Orchestrion each note on a marimba can be played simultaneously or sequentially in rapid fashion because a solenoid is placed on each key. This approach allows for increased flexibility in choosing different types of push and pull solenoids that can be used to achieve increased speed and dynamic variability.¹⁸ Unlike those of human percussionists, the single strokes generated by solenoids or motors can move at such great speeds it is not necessary to create double-stroke functionality.

While solenoid-based designs are

mechanically simple and capable of a wide range of expression, they tend to lack useful visual cues. Using a human shape provides interacting musicians and audience members with a framework they are more familiar with, thus, allowing them to better understand the relationship between sound generation and the accompanying robotic movements and gestures. Additionally, the visual cues provided by the larger human-like motion and gestural movements of humanoid robots allow the musicians to anticipate the robot's actions and improve synchronization. Such visual cues cannot be yielded by a system such as Orchestrion. Some robots combine functions of both design methods in an attempt to simultaneously gain the benefits of the inhuman sound producing capabilities and important visual cues. The marimba-playing robot, Shimon, has four arms each with two solenoid activators for striking the keys. The movements of the arms combined with the eight fast-moving solenoids provides advantages from both design methods.³⁸



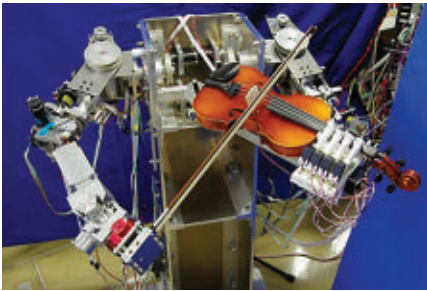
Shimon from Georgia Tech's Center for Music Technology.

Stringed instruments. Similarly to the percussive robots, the design of several string-playing robots focus on achieving simple, yet robust excitation and pitch manipulation rather than mimetic control and anthropomorphism. Sergi Jordà's Afasia system¹⁴ is one such example. In this system solenoids are used to pluck strings, while pitch is controlled by either push solenoids placed along the bridge or motors that move the bridge itself. Singer's Guitar-Bot uses a wheel embedded with guitar picks that can rotate at variable speeds to pluck strings and relies on a moving bridge assembly that rides on pulleys driven by a DC motor.

Anthropomorphism and emulating human plucking motion has not been attempted until recently. Work

^e <http://www.theorchestrionproject.com/>

by Chadeaux et al. involves motion capture of human harp playing and the design of a robotic harp-playing finger. Silicone of various shape and hardness is placed on the fingertips in order to replicate different vibrational characteristics.⁶ An anthropomorphic violin-bowing arm was developed by Shibuya et al. The shoulder, elbow, and wrist is modeled to create expressive playing by controlling the bow's pressure, position, and rate of movement.³² Similarly, Toyota's robotic violin player replicates human form and motion.^f

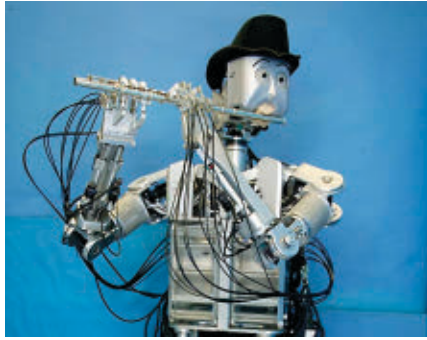


Violin-playing robot from Ryukoku University.

Wind instruments. A main challenge for wind-instrument playing machines is producing the necessary energy fluctuations in order to make the instrument resonate at desired frequencies and volumes. Some address this challenge with non-humanoid designs and function. Roger Dannenberg's McBlare is a non-humanoid robotic bagpipe player. An air compressor provides air and electro-magnetic devices control the power to "fingers," which open and close the sound holes. While the speed of McBlare's control mechanisms surpasses that of expert bagpipe performers it cannot equal the human skill of adapting to the characteristics of the reed.¹⁰ Another example of a non-anthropomorphic robotic wind instrument is the Logo's Foundation's Autosax,²⁴ which uses "compression drivers and acoustic impedance converters that feed the drive signal to the instrument via a capillary."

Other wind instrument designs focus on replicating and understanding human physiology and anatomy. Ferrand and Vergéz designed a blowing machine to examine the nonlinear dynamics natural human airflow by mod-

eling the human mouth.¹¹ They found the system's performance was affected by the servo valve's friction, cut-off frequency, and mounting stiffness of the artificial lips. Waseda University developed a humanoid flute-playing robot and more recently a sax-playing robot.³⁶ The systems include degrees-of-freedom (DOFs) for controlling finger dexterity, lip and tongue control, and lung control (airpump and valve control).



Flutist from Waseda University.

Augmenting traditional instruments and new instruments and interfaces. Unlike the examples described previously that demonstrate traditional instruments being played in traditional ways, there are also examples of traditional instruments being played in non-traditional ways. Augmenting musical instruments with mechanics is useful for acquiring additional sonic variety and playability. Trimpin's Contraption Instant Prepared Piano is an example of instrument augmentation and enables mechanical bowing, plucking, and vibrating of piano strings to increase the harmonic and timbral range of the instrument. In many augmented instruments the human performer and robotic mechanisms share control of the sound producing characteristics. This creates an interesting interactive scenario that combines the traits inherent to the interaction of two individual musicians and the interaction between performer and instrument. Richard Logan-Greene's systems, Brainstem Cello and Actionclarinet,⁸ explore this type of shared interactive control as the instruments are outfitted with sensors to respond to the human performer by using mechanics to dampen strings or manipulate pitch in real time.

There are also examples of com-

pletely new instruments in which automated mechanical functions are necessary to create a performance. Trimpin's "Conloninpurple" installation is made up of marimba-like bars coupled with aluminum resonating tubes and solenoid triggers. Each key is dispersed throughout a room to create a massive instrument capable of "three-dimensional" sound. Logan-Greene's Submersible is a mechanical instrument that raises and lowers pipes in and out of water to modify their pitch and timbre.²³



Submersible by Richard Logan-Greene.

Musical Intelligence

The ultimate goal of robotic musicianship is to create robots that can demonstrate musicality, expressivity, and artistry while stimulating innovation and newfound creativity in other interacting musicians. In order to achieve such artistic sophistication a robotic musician must have the ability to extract information relevant to the music or performance and apply this information to its underlying decision processes.

Sensing methods. Providing machines with the ability to sense and reason about the world and their environments is a chief objective of robotics research in general. In musical applications, naturally, a large emphasis of sensing and analysis lies within the auditory domain. Extracting meaningful information from an audio signal is referred to as machine listening.³¹ There are numerous features that can be used to describe the characteristics of a musical signal. These descriptors are hierarchical in terms of both complexity and how closely they resemble human perceptions. Lower-level features typically describe physical characteristics of the signal. Higher-level descriptors, though often more difficult to compute, tend to correlate more strongly with the terms people generally use to describe music such as mood, emotion, and genre. Often

^f http://www.toyota-global.com/innovation/partner_robot/

^g <http://zownts.com/>

these higher-level descriptors are classified by using a combination of low-level (amplitude and spectral centroid) and mid-level (tempo, chord, and timbre) features.

Determining which attributes of the sound are important is decided by the developer and predominantly relies on the desired functionality of the robot and its interaction with a person. Specific characteristics of the sound might be required for aesthetic reasons to support a specific interaction or there may be something inherent to the nature of the music that calls for a more narrowed focus in its analysis. For example, perhaps the developer intends for the system to interact with a drummer. In such a scenario audio analysis techniques that focus on rhythmic and timing qualities might be preferred over an analysis of pitch and harmony.

In fact, for many of the drumming robots the audio processing algorithms are indeed focused on aspects related to timing. A first step in this analysis is to detect musical events such as individual notes. This is referred to as onset detection and many methods exist for measuring an onset based on the derivative of energy with respect to time, spectral flux, and phase deviation.⁴ Instruments that exhibit longer transients (such as a cello or flute) and certain playing techniques (such as vibrato and glissandi) can pose challenges in detection. It is also controvertible whether the system should detect all physical onsets or all humanly perceptible onsets. Some methods are psycho-acoustically informed to support the natural human perceptions in different portions of the frequency spectrum.⁹

An analysis of the onsets over time can provide additional cues related to note density, repetition, and beat. The notion of beat, which corresponds to the rate at which humans may tap their feet to the music, is characteristic to many types music. It is often computed by finding the fundamental periodicity of perceived onsets using autocorrelation.

The *Haile* robot continuously estimates the beat of the input rhythms and uses the information to inform its response.³⁸ In Barton's HARMi (Human and Robot Musical Improvisation) software he explores rhythmic texture by detecting multiple beat rates and not just those most salient to human listen-



The ultimate goal of robotic musicianship is to create robots that can demonstrate musicality, expressivity, and artistry while stimulating innovation and newfound creativity in other interacting musicians.



ers. This allows for more expressivity in the system by increasing rhythmic and temporal richness and allowing the robot to “feel” the beat in different ways. As a result the robot can detect “swing” in jazz music and the more rubato nature intrinsic to classical music.³

Other analysis methods help robots to perceive the pitch and harmonic content of music. Though pitch detection of a monophonic signal has been achieved with relatively robust and accurate results, polyphonic pitch detection is more challenging.²¹ It is often desired to extract the melody from a polyphonic source. This generally consists of finding the most salient pitches and using musical heuristics (such as the tendency of melodic contours to be smooth) to estimate the most appropriate trajectory.²⁹

Detecting harmonic structure in the context of a chord (three or more notes played simultaneously) is also important. Chord detection and labeling is performed using “chroma” features. The entire spectra is projected into 12 bins representing the 12 notes (or chroma) of a musical octave. The distribution of energy over the chroma is indicative of the chord being played. In Western tonal music Hidden Markov Models (HMMs) are often useful because specific chord sequences tend to occur more often and applying these chord progression likelihoods can be beneficial for chord recognition and automatic composition.⁷

Pitch tracking and chord recognition are useful abilities for robotic musicians. Some robots rely on these pitch-based cues to perform score following. The robot listens to other musicians in an ensemble and attempts to synchronize with them by estimating their location within a piece.^{26,34} Otsuka et al. describe a system that switches between modules based on melodic and rhythmic synchronization depending on a measure of the reliability of the melody detection.²⁸ Other robotic systems that improvise such as Shimon use the chords in the music as a basis for its note generative decisions.

As an alternative to utilizing audio analysis for machine listening, symbolic protocols such as Musical Instrument Digital Interface (MIDI) and Open Sound Control (OSC) can be used to simplify some of the challenges in-

herent to signal analysis.

Machine vision and other sensing methods. Other cues outside of the audio domain can also provide meaningful information during a musical interaction. A performer's movement, posture, and physical playing technique can describe information related to timing, emotion, or even the amount of experience with the instrument. Incorporating additional sensors can improve the machine's perceptive capabilities, enabling it to better interpret the musical experience and perceptions of the interacting musicians.

Computer vision techniques have proven to be advantageous for several musical applications and interaction scenarios. Pan et al. program their humanoid marimba player to detect the head nods of an interacting musician.³⁰ The nods encourage a more natural communication so humans and robots can exchange roles during the performance. Solis' anthropomorphic flutist robot similarly uses vision to create more instinctive interactions by detecting the presence of a musical partner. Once detected the robot then listens and evaluates its partner's playing.³⁴ Shimon uses a Microsoft Kinect to follow a person's arm movements. A person controls the specific motifs played by the robot and the dynamics and tempos by moving his arms in three dimensions. Shimon was also programmed to track infra-red (IR) lights mounted to the end of mallets. A percussionist is able to train the system with specific gestures and use these gestures to cue different sections of a precomposed piece.⁵

Mizumoto and Lim incorporate vision techniques to detect gestures for improved beat detection and synchronization.^{22,26} A robotic theremin player follows a score and synchronizes with an ensemble by using a combination of auditory and visual cues. Similarly, the MahaDeviBot utilizes a multimodal sensing technique to improve tempo and beat detection. Instead of using vision, an instrumentalist wears sensors measuring acceleration and orientation to provide information regarding his or her movement. The combined onsets detected by the microphone and wearable sensors are processed by Kalman filtering and the robot detects



Incorporating additional sensors can improve the machine's perceptive capabilities, enabling it to better interpret the musical experience and perceptions of the interacting musicians.



tempo in real time.¹⁶

Hochenbaum and Kapur extended their multimodal sensing techniques to classify different types of drum strokes and extract other meaningful information of a performance.¹² They placed accelerometers on drummers' hands to automatically annotate and train a neural network to recognize the different drum strokes from audio data. They then examined the inter-onset-intervals between onsets found in both the audio and accelerometer data. They found there was a greater consistency between the physical and auditory onsets for the more expert musician. Such multimodal analyses that combine the physical and auditory spaces are becoming more standard in performance analysis.

The ability to synchronize musical events between interacting musicians is very important. For synchronization to occur, a system must know an event will occur before it actually happens. In music, these events are often realized in the form of a musical score, however, in some interactions such as improvisation there is no predefined score and synchronization may still be desired. Cicconet et al. explored visual cues based anticipation by developing a vision system that predicts the time at which a drummer will play a specific drum based off of his movement.⁸ Shimon was programmed to use these predictions to play simultaneously with drummer. The anticipation vision analyzes the up and downward motion of the drummer's gesture by tracking an IR sensor attached to the end of a mallet. Using the mallet's velocity, acceleration, and location the system predicts when the drumstick will make contact with a particular drum.

Generative functions. The second aspect of musical intelligence focuses on the system's ability to play and perform music. This includes functions such as reading from a score, improvising, or providing accompaniment. While many examples for this functionality exist in the domain of computer-generated music, here we focus on methods that have been implemented specifically for robotic musicians. These generative functions may rely on statistical models, predefined rules, or abstract algorithms.

Modeling human performers. When

reading from a precomposed score the robot does not need to decide what notes to play, but instead must decide how to play them in order to create an expressive performance. Designing expressivity in a robot is often achieved by observing human performers. Solis et al.³⁵ describe such a methodology in their system. The expression of a professional flute player is modeled using artificial neural networks (ANNs) by extracting various musical features (pitch, tempo, volume, among others) from the performance. The networks learn the human's rules for expressivity and output parameters describing a note's duration, vibrato, attack time, and tonguing. These parameters are then sent to the robot's control system.

Modeling human performers can also be used for improvisational systems. Shimon generates notes using Markov decision processes trained on performances of jazz greats such as John Coltrane and Thelonious Monk.²⁷ The pitch and duration of their notes are modeled in 2nd order Markov chains.

Kapur et al. developed a robotic harmonium called Kritaanjli, which extracts information from an interacting human performer's style of harmonium pumping and attempts to emulate it.¹⁷ The robot's motors use information from a linear displacement sensor that measures the human's pump displacement.

Algorithmic composition. Some music artificial intelligence (AI) systems rely on sets of rules, a hybrid of statistics and rules, or even algorithms with no immediate musical relevance to drive their decision processes.

The rule-based systems may attempt to formulate music theory and structure into formal grammars or may creatively map the robot's perceptual modules to its sound generating modules. The Man and Machine Robot Orchestra at Logos has performed several compositions which employ interesting mappings²⁴ for musical expression. In the piece, *Horizon's for Three* by Kristof Lauwers and Moniek Darge, the pitches detected from a performer playing an electric violin are directly mapped to pitches generated by the automatons. The movements of the performers control the wind pressure of the organ. In another piece, *Hyperfolly* by Yvan Vander Sanden, the artist presses but-

tons that trigger responses from the machines. Pat Metheny similarly controls the machines in his *Orchestrion* setup. By playing specific notes or using other triggers the machines play precomposed sections of a piece.

For some interactions the robots are programmed with short musical phrases and rules for modifying and triggering them are implemented. Kritaanjli is programmed with a database of ragas from Indian classical music. The robot accompanies an improvising musician by providing the underlying melodic content of each raga and uses additional user input to switch between or modify the ragas.¹⁷ Shimon similarly uses predefined melodic phrases in a piece called *Bafana* by Gil Weinberg inspired by African marimba ensembles. Shimon listens for specific motifs and depending on the mode may either mimic and synchronize or play in canon creating new types of harmonies. At another point in the piece Shimon begins to change motifs using stochastic decision processes and the human performer then has a chance to respond.⁵

Many interaction scenarios designed for robotic musicians employ a hybrid of statistical models and rules. Different modes may rely on different note generative techniques. These modes can be mapped to certain parameters and triggered by interacting musicians either through the music (by playing a specific phrase or chord), physical motion (completing a predefined gesture) or other sensor (light, pressure, and so on). Cues that signal different sections or behaviors are commonly used in all-human ensembles and the question in robotic musicianship is how to create cues that can be robustly detected by the AI, yet, still maintain a natural and musical distinction.

Miranda and Tikhonoff describe an autonomous music composition system for the Sony AIBO robot that employs statistical models as well as rules.²⁵ The system is trained on various styles of music and a short phrase is generated based off of the previous chord and first melodic note of the previous phrase. The AIBO interacts with its environment and the behavior of the composition is modified based off of obstacles, colors, the presence of humans, and different emotional states.

Though using traditional AI techniques such as ANNs and Markov Chains might capture the compositional strategies of humans, it is also common for composers and researchers to adapt a wide variety of algorithms to music generating functions. Often the behaviors of certain computational phenomena can yield interesting and unique musical results. Chaos, cellular automata, genetic, and swarm algorithms have all been repurposed for musical contexts and a few specific to robotic musicianship performances.

Weinberg et al. use a real-time genetic algorithm to establish human-robot improvisation.³⁹ Short melodic phrases form the initial base population and dynamic time warping (DTW) is used as a similarity metric between observed phrases of the human performer in the algorithms fitness function. Mutating the phrases allows for more complexity and richer harmony and melodic content by transposing, inverting, and adding semitones to the phrases. The algorithm was implemented in *Haile* and the performance uses a call and response methodology in which the human performer and robot take turns using each other's music to in their own responses.

Exploring decentralized musical systems has been explored by Albin et al. using musical swarm robots.¹ Several mobile robots are each equipped with a smart-phone and solenoid that strikes a pitched bell. The robot's location is mapped to different playing behaviors that manipulate the dynamics and rhythmic patterns played on the bell. Unique rhythmic and melodic phrases emerge when the robots undergo different swarming behaviors such as predator-prey, follow-the-leader, and flocking. Additionally, people interact with the robots by sharing the space with the robots and the robots respond to their presence (using face detection on the phone).

Embodiment

By integrating robotic platforms with machine musicians, researchers can bring together the computational, sonic, visual, and physical worlds, creating a sense of embodiment—the representation or expression of objects in a tangible and visible form. This can lead to richer musical interactions as users demonstrate increased attention, an ability to distinguish between an AI's

COMMUNICATIONS APPS

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



Available for iPad, iPhone, and Android



Available for iOS, Android, and Windows

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



Association for Computing Machinery

pragmatic actions (physical movements performed to bring the robot closer to its goal such as positioning its limbs to play a note) and epistemic actions (physical movements performed to gather information or facilitate computational processing such as positioning its sensors for an improved signal-to-noise ratio), and better musical coordination and synchronization as a result of the projection of physical presence and gestures.

Several human-robotic interaction studies outside of the context of music have demonstrated desirable interaction effects as a result of physical presence. The studies have found physical embodiment leads to increased levels of engagement, trust, perceived credibility, and more enjoyable interactions.²⁰ Robotic musicians yield the same benefits for those directly involved with the interaction (other musicians) as well those merely witnessing it (audience members).

Physical presence can also directly affect the quality of the music. Hoffman and Weinberg examined the effects of physical presence and acoustic sound generation on rhythmic coordination and synchronization.¹³ The natural coupling between the robot's spatial movements and sound generation afforded by embodiment allows for interacting musicians to anticipate and coordinate their playing with the robot, leading to richer musical experiences.

In addition to the cues provided by a robotic musician's sound producing movements, physical sound-accompanying gestures can create better informed social interactions by communicating information related to a task, an interaction, or a system's state of being. An important emphasis in our own research has been the design and development of robotic physical gestures that are not directly related to sound production, but can help convey useful musical information for human musicians and an observing audience. These gestures and physical behaviors are used to express the underlying computational processes allowing observers to better understand how the robot is perceiving and responding to the music and the environment. A robot's gaze, posture, velocity, and motion trajectories can be manipulated for communicative function. For example, simple behaviors designed for

Shimon, such as nodding to the beat, looking at its marimba during improvisation, and looking at interacting musicians when listening help enhance synchronization among the ensemble members and to reinforce the tempo and groove of the music.

In robotics, the term "path planning" refers to the process of developing a plan of discrete movements that satisfies a set of constraints to reach a desired location. In a musical context, robots may not be able to perform specific instructions in a timely, coordinated, and expressive manner, which requires employing musical considerations in designing a path-planning algorithm. Such considerations may shed light on conscious as well as subconscious path planning decisions human musicians employ while coordinating their gestures (such as finger order in piano playing). In robotic musicianship path planning and note generative and rhythmic decisions become an integrated process.

Summary and Evaluation

In this article we presented several significant works in the field of robotic musicianship and discussed some of the different methodologies and challenges inherent to designing mechanical systems that can reason about music and musical interactivity. The projects discussed bring together a wide variety of disciplines from mechanics, through perception, to artificial creativity. Each system features a unique balance between these disciplines, as each is motivated by a unique mixture of goals and challenges, from the artistic, through scientific, to the engineering-driven. This interdisciplinary set of goals and motivations makes it difficult to develop a standard for evaluating robotic musicians. For example, when the goal of a project is to design a robot that can improvise like a jazz master, then a musical Turing test can be used for evaluation; if the goal is artistic and experimental in nature, such as engaging listeners with an innovative performance, then perhaps positive reviews of the concert are indicative of success; and if the project's motivation is to study and model humans' sound generation mechanism, anatomy- and cognitive-based evaluation methods can be employed.

Most of the projects discussed in this article include evaluation methods that address the project's specific goals and motivations. Some projects, notably those with artistic and musical motivations, do not include an evaluation process. Others address multiple goals, and therefore include a number of different evaluation methods for each aspect of the project. For example, in computational-cognition based projects, robots' perceptual capabilities are measured by comparing the algorithm against a ground truth and calculating accuracy. However, establishing ground truths in such projects has proven to be a challenging task as many parameters in music are ambiguous and subject to individual interpretation. HRI motivated experiments, on the other hand, are designed to evaluate interaction parameters such as the robots' ability to communicate and convey relevant information through their music generating functions or physical gesture. We believe as research and innovation in robotic musicianship continues to grow and mature, more unified and complete evaluation methods will be developed.

Though robotic musicianship has seen many great developments in the last decade, there are still many white spaces to be explored and challenges to be addressed—from developing more compelling generative algorithms to designing better sensing methods enabling the robot's interpretations of music to correlate more closely with human perceptions. We are currently developing new types of robotic interfaces, such as companion robots and wearable robotic musicians, with the goal of expanding the breadth and depth of musical human-robotic interaction. In addition to developing new interfaces, sensing methods, and generative algorithms we are also excited to showcase our developments through performances as part of the Shimon Robot and Friends Tour.^h We look forward to continue to observe and contribute to new developments in this field, as robotic musicians continue to increase their presence in everyday life, paving the way to the creation of novel musical interactions and outcomes. ■

^h <http://www.shimonrobot.com>

References

- Albin, A., Weinberg, G. and Egerstedt, M. Musical abstractions in distributed multi-robot systems. In *Proceeding of the International Conference on Intelligent Robots and Systems* (2012). IEEE/RSJ, 451–458.
- Atkeson, C.G. et al. Using humanoid robots to study human behavior. *Intelligent Systems and their Applications* 15, 4 (2000). IEEE, 46–56.
- Barton, S. The human, the mechanical, and the spaces in between: Explorations in human-robotic musical improvisation. In *Proceeding of the 9th Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.
- Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M. and Sandler, M.B. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing* 13, 5 (2005), 1035–1047.
- Bretan, M., Cicconet, M., Nikolaidis, R. and Weinberg, G. Developing and composing for a robotic musician. In *Proceeding of the International Computer Music Conference* (Ljubljana, Slovenia, Sept. 2012).
- Chadefaux, D., Le Carrou, J.-L., Vitran, M.-A., Billout, S. and Quartier, L. Harp plucking robotic finger. In *Proceeding of the International Conference on Intelligent Robots and Systems* (2012) IEEE/RSJ, 4886–4891.
- Chen, R., Shen, W., Srinivasamurthy, A. and Chordia, P. Chord recognition using duration-explicit hidden markov models. In *Proceedings of ISMIR* (2012), 445–450.
- Cicconet, M., Bretan, M. and Weinberg, G. Human-robot percussion ensemble: Anticipation on the basis of visual cues. *Robotics & Automation Magazine* 20, 4 (2013). IEEE, 105–110.
- Collins, N.M. Towards autonomous agents for live computer music: Real-time machine listening and interactive music systems. Ph.D. thesis, Citeseer, 2006.
- Dannenber, R.B., Brown, H.B. and Lupish, R. Mcblare: A robotic bagpipe player. In *Musical Robots and Interactive Multimodal Systems*. Springer, 2011, 165–178.
- Ferrand, D. and Vergez, C. Blowing machine for wind musical instrument: toward a real-time control of the blowing pressure. In *Proceedings of the 16th Mediterranean Conference on Control and Automation*. IEEE, 2008, 1562–1567.
- Hochenbaum, J. and Kapur, K. Drum stroke computing: Multimodal signal processing for drum stroke identification and performance metrics. In *Proceedings of the 2012 International Conference on New Interfaces for Musical Expression*.
- Hoffman, G. and Weinberg, G. Synchronization in human-robot musicianship. In *Proceeding of 2010 RO-MAN*. IEEE, 718–724.
- Jordà, S. Afasia: the ultimate Homeric one-man-multimedia-band. In *Proceedings of the 2002 Conference on New Interfaces for Musical Expression*. National University of Singapore, 1–6.
- Kapur, A. A history of robotic musical instruments. In *Proceedings of the 2005 International Computer Music Conference*, 21–28.
- Kapur, A. Multimodal techniques for human/robot interaction. *Musical Robots and Interactive Multimodal Systems*. Springer, 2011, 215–232.
- Kapur, A., Murphy, J. and Carnegie, D. Kritaanjli: A robotic harmonium for performance, pedagogy and research.
- Kapur, A., Trimpin, E.S., Suleman, A. and Tzanetakis, G. A comparison of solenoid-based strategies for robotic drumming. In *Proceedings of 2007 ICMC* (Copenhagen, Denmark, 2007).
- I. Kato, I., Ohteru, S., Shirai, K., Matsushima, T., Narita, S., Sugano, S., Kobayashi, T. and Fujisawa, E. The robot musician "wobot-2" (waseda robot-2). *Robotics* 3, 2 (1987), 143–155.
- Kidd, C.D. and Breazeal, C. Effect of a robot on user perceptions. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3559–3564.
- Lerch, A. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, 2012.
- Lim, A., Mizumoto, T., Cahier, L.-K., Otsuka, T., Takahashi, T., Komatani, K., Ogata, T. and Okuno, H.G. Robot musical accompaniment: Integrating audio and visual cues for real-time synchronization with a human flutist. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1964–1969.
- Logan-Greene, R. *Submersions I*. University of Washington, 2011.
- Maes, L., Raes, G.-W. and Rogers, T. The man and machine robot orchestra at logos. *Computer Music Journal* 35, 4 (2011), 28–48.
- Miranda, E.R. and Tikhonoff, V. Musical composition by autonomous robots: A case study with aibo. In *Proceedings of 2005 Conference Towards Autonomous Robotic Systems*.
- Mizumoto, T., Lim, A., Otsuka, T., Nakadai, K., Takahashi, T., Ogata, T. and Okuno, H.G. Integration of flutist gesture recognition and beat tracking for human-robot ensemble. In *Proceedings of IEEE/RSJ-2010 Workshop on Robots and Musical Expression*, 159–171.
- Nikolaidis, R. and Weinberg, G. Playing with the masters: A model for improvisatory musical interaction between robots and humans. In *Proceedings of RO-MAN 2010*. IEEE, 210–217.
- Otsuka, T., Nakadai, K., Takahashi, T., Ogata, T. and Okuno, H.G. Real-time audio-to-score alignment using particle filter for coplayer music robots. *EURASIP J. Advances in Signal Processing* 2 (2011).
- Paiva, R.P., Mendes, T. and Cardoso, A. On the detection of melody notes in polyphonic audio. In *Proceedings of 2005 ISMIR*, 175–182.
- Pan, Y., Kim, M.-G. and Suzuki, K. A robot musician interacting with a human partner through initiative exchange. In *Proceedings of the 2010 Conference on New Interfaces for Musical Expression*, 166–169.
- Rowe, R. *Machine Musicianship*. The MIT Press, 2004.
- Shibuya, K., Ideguchi, H. and Ikushima, K. Volume control by adjusting wrist moment of violin-playing robot. *International J. Synthetic Emotions* 3, 2 (2012), 31–47.
- Singer, E., Feddersen, J., Redmon, C. and Bowen, B. Lemur's musical robots. In *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*. National University of Singapore, 181–184.
- Solis, J., Chida, K., Isoda, S., Suefuiji, K., Arino, C. and Takanishi, A. The anthropomorphic flutist robot wf-4r: From mechanical to perceptual improvements. In *Proceeding of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 64–69.
- Solis, J., Suefuiji, K., Taniguchi, K., Ninomiya, T., Maeda, M. and Takanishi, A. Implementation of expressive performance rules on the wf-4r-ii by modeling a professional flutist performance using nn. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, 2552–2557.
- Solis, J., Takanishi, A. and Hashimoto, K. Development of an anthropomorphic saxophone-playing robot. *Brain, Body and Machine*. Springer, 2010, 175–186.
- Weinberg, G. and Driscoll, S. Toward robotic musicianship. *Computer Music Journal* 30, 4 (2006), 28–45.
- Weinberg, G. and Driscoll, S. The design of a perceptual and improvisational robotic marimba player. In *Proceedings of the 16th IEEE International Symposium on Robot and Human Interactive Communication*, 2007, 769–774.
- Weinberg, G., Godfrey, M., Rae, A. and Rhoads, J. A real-time genetic algorithm in human-robot musical improvisation. *Computer Music Modeling and Retrieval. Sense of Sounds*. Springer, 2008, 351–359.
- Williamson, M.M. Robot arm control exploiting natural dynamics. Ph.D. thesis. Massachusetts Institute of Technology, 1999.
- Zhang, A., Malhotra, M. and Matsuoka, Y. Musical piano performance by the act hand. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 2011, 3536–3541.

Mason Bretan (pbretan3@gatech.edu) is a Ph.D. candidate in the robotic musicianship group at Georgia Institute of Technology, Atlanta.

Gil Weinberg (gilw@gatech.edu) is the director of the Georgia Tech Center for Music Technology at Georgia Institute of Technology, Atlanta.

© 2016 ACM 0001-0782/16/05 \$15.00



Watch the authors discuss their work in this exclusive *Communications* video. <http://cacm.acm.org/videos/a-survey-of-robotic-musicianship>

Systor2016

**The 9th ACM International
Systems and Storage Conference**

June 6 – 8 • Haifa, Israel

We invite you to attend SYSTOR 2016, an ACM SIGOPS Systems and Storage Conference covering all aspects of systems research. Join us in Haifa from June 6 to June 8 for an exciting and enriching technical program, social events, and informal mingling with leading researchers, from both academia and industry.

Submission is still open for posters and highlight papers, so don't miss the opportunity to feature your work!

Registration is free of charge, and will be open in the near future.

Highlight paper submission deadline: May 6, 2016

Poster submission deadline: April 8, 2016

Program chairs

Mark Silberstein, Technion
Emmett Witchel, University of Texas

General chair

Katherine Barabash, IBM Research

Posters chair

Anna Levin, IBM Research

Steering committee head

Michael Factor, IBM Research

Steering committee

Ethan Miller, University of California
Santa Cruz

Liuba Shrira, Brandeis University

Dan Tsafir, Technion

Dalit Naor, IBM Research

Erez Zadok, Stony Brook University

www.systor.org/2016/

Sponsored by



In cooperation with



Platinum sponsor



Gold sponsors



Sponsors



research highlights

P. 112

**Technical
Perspective**
**A Breakthrough
in Software
Obfuscation**

By Boaz Barak

P. 113

Hiding Secrets in Software: A Cryptographic Approach to Program Obfuscation

By Sanjam Garg, Craig Gentry, Shai Halevi,
Mariana Raykova, Amit Sahai, and Brent Waters

P. 121

**Technical
Perspective**
Software Is Natural

By Gail C. Murphy

P. 122

On the Naturalness of Software

By Abram Hindle, Earl T. Barr, Mark Gabel,
Zhendong Su, and Premkumar Devanbu

Technical Perspective

A Breakthrough in Software Obfuscation

By Boaz Barak

FOR THOUSANDS OF years, cryptography was synonymous with “secret writing” whereby two parties used some shared secret key or method to communicate confidential information. But beginning in the 1970s, there was an explosion of new cryptographic concepts and applications. These include *public key cryptography*—confidential communication between two parties over an open channel, *secure function evaluation*—jointly computing a function of private inputs (such as the results of an election or auction) without revealing them, *zero knowledge proofs*—proving a statement is true without revealing anything else, *fully homomorphic encryption*—computing on encrypted data, and more.

These notions seem so paradoxical it is amazing these cryptography pioneers even *imagined* they could ever be achieved!^a Based on their writing, it seems at least part of these inventors’ thought process involved the following mental experiment: it not too difficult to convince yourself these wonderful objects can in fact be achieved if we had a *black box* computing some function, whereas every party could use the box to compute an output from an input but cannot understand its internal working. In the words of James Ellis,^b “we can regard our [encryption function] as a look-up table containing one value of output for each possible input value.” The hope was it would be possible to simulate such a program by using what Diffie and Hellman called a “one-way compiler, which takes

an easily understood program in a high-level language and translates it into an incomprehensible program in some machine language.”^c

Such a compiler is known as a *software obfuscator*, and despite serving as a guiding light as to what cryptographic tools we can hope to achieve, in the many years since, most practical and theoretical results on this notion have been *negative*. Obfuscation was and still is practiced to protect intellectual property and other secrets in source code, but no practical scheme has been found that resists a truly determined attacker. On the theory front, in 2001 my co-authors and I showed that for arguably the most natural notion of obfuscation, known as “virtual black box” obfuscation, is *impossible* to achieve.^d We left a “tiny” loophole by noting it did not rule out a weaker notion known as “indistinguishability obfuscation” or IO. However, at least to me it seemed this notion of IO may well be also impossible to achieve and even if it was achievable, it did not seem very useful. The following paper of Garg et al. proved I was probably wrong on the first count and definitely wrong on the second one.

In this remarkable work, the authors construct a “one-way compiler” of the type envisioned by Diffie and Hellman. This is a way to transform a program P into a program Q that is functionally equivalent, in the sense it would produce the same output as P on every input, but is “incomprehensible” in the sense it satisfies this IO condition. The compiler itself is very different than all prior obfuscators. It is based on another breakthrough paper by a subset of these

authors,^e who constructed an object known as “cryptographic multilinear maps.” They combine this with a result of Barrington^f that originally arose in the context of circuit lower bounds. The resulting compiler can be thought of as somewhat analogous to converting the program P into a scrambled Rubik’s cube. If you interpret the input x as a sequence of rotations to apply to the cube, the final pattern will convey the output $P(x)$. But you will not be able to learn any other information from it.

The second contribution of this paper was to show this seemingly weak notion of IO is in fact sufficiently strong to obtain some of the most exciting applications of obfuscation. This has been greatly extended in many follow-up works, which have established IO as a kind of a “cryptographic master tool” that can yield a great many other applications.

This paper opens many more research directions than it closes. For starters, while the authors gave some evidence for the security of their construction, it is not truly satisfying, and basing IO on firmer foundations is a crucial research objective. Also, at the moment the construction is terribly impractical, incurring a huge blowup to even the simplest programs. Both of these and more are now the object of intensive research efforts. It truly is an exciting time for cryptography. 

a Indeed, when Ralph Merkle, who was one of the first people to conceive of the idea of public-key encryption, along with Whit Diffie and Martin Hellman, as well as James Ellis of the British agency CESG, submitted his work to *Communications of the ACM*, he noted the only reference he could find to this problem was in a science fiction story.

b J. Ellis. The possibility of secure non-secret digital encryption. *CESG Report*, Jan. 1970

c W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory* 22, 6 (1976), 644–654.

d B. Barak et al. On the (im)possibility of obfuscating programs. *Advances in Cryptology*. Springer Berlin Heidelberg, 2001, 1–18. Also, *JACM* 2012.

e S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. *Eurocrypt* 7881 (2013), 1–17.

f D. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC1. *J. Comput. Syst. Sci.* 38, 1 (1989), 150–164.

Boaz Barak (info@boazbarak.org) is the Gordon-McKay Professor of Computer Science at the Harvard University, Cambridge, MA.

Hiding Secrets in Software: A Cryptographic Approach to Program Obfuscation

By Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters

Abstract

Can we hide secrets in software? Can we obfuscate programs—that is, make programs *unintelligible* while preserving their functionality? What exactly do we mean by “unintelligible”? Why would we even want to do this?

In this article, we describe some rigorous cryptographic answers to these quasi-philosophical questions. We also discuss our recent “candidate indistinguishability obfuscation” scheme and its implications.

1. INTRODUCTION

Informally, the goal of program obfuscation is deceptively easy to state: to make a program unintelligible while preserving its functionality. By “preserving its functionality,” we just mean that the obfuscated program should be fully executable and have the same input-output behavior as the original program. Defining “unintelligible” is more tricky.

1.1. Making sense of the “unintelligible”

Of course, most program code is already “unintelligible” in the colloquial sense of the word. “Indeed, any programmer knows that total unintelligibility is the natural state of computer programs (and one must work hard in order to keep a program from deteriorating into this state).”¹

But suppose we want a stronger notion of unintelligibility, one that applies not only to sleep-deprived programmers but also to automated “deobfuscators” and determined, resourceful adversaries. Previous work on obfuscation has focused on making code resistant/unintelligible to particular attacks, such as known techniques for static and dynamic analysis.⁶ However, it is a recurring theme in security engineering that resistance to one set of attacks is no guarantee that a system cannot be broken by other means. And so, many of these obfuscation schemes have been followed by corresponding schemes for deobfuscation.⁶

Modern cryptography, especially following the work of Goldwasser and Micali,¹⁸ has offered a rigorous and expansive approach to defining security, offering the best way to escape the break-and-repair cycle of security engineering: Define security (in our case, the word “unintelligible”) in a *mathematically rigorous* way, and then *prove* that if there exists *any* adversary that can efficiently break security (according to the definition), then there is an efficient algorithm to solve a “hard” mathematical problem, such as factoring large integers. This approach makes a *computational assumption*—that the underlying mathematical problem is hard (i.e., cannot

be solved in time polynomial in a security parameter), and that the adversary is computationally bounded to run in polynomial time. Under this computational assumption the scheme is *provably secure*. Crucially, this approach makes no other assumption or guess about what type of attack the adversary might deploy to break security. Of course, the security definition might not completely capture the adversary’s access to the system: for example, early definitions of security for encryption assumed the secret key could be stored and used securely, while “side channel” attacks have challenged this assumption and led to refined definitions. And the “hard” problem might turn out not to be as hard as initially thought: for example, algorithms for factoring large integers have improved immensely since the invention of RSA. But history has shown that the rigorous cryptographic approach to security is the best way to filter out schemes that break in “uninteresting” ways.

So, how does cryptography rigorously define the unintelligibility of obfuscation? The natural conceptual way to view the unintelligibility of obfuscation is as a *black box* that allows oracle access to a program (you give it inputs and get back outputs) but which completely hides the program’s internal workings. Hada²⁰ and Barak et al.¹ were able to make this definition of obfuscation mathematically rigorous, but then Barak et al. showed that even a weaker notion called *virtual black box* (VBB) obfuscation is impossible to achieve: some programs are VBB-unobfuscatable. Even stronger negative results were obtained for other models.¹⁷ This profound negative result, and our inability to find a clear line between obfuscatable and unobfuscatable programs, essentially crushed cryptographic research on obfuscation for over a decade.

Barak et al.¹ also presented a second way of viewing program obfuscation: as program *pseudo-canonicalization*. The idea is that if programs P_0 and P_1 have the *same* input-output behavior, and I obfuscate one of them, it should be hard for an adversary to tell which program I obfuscated. Thus, the obfuscation “canonicalizes” functionally equivalent programs by making them hard to tell apart. This notion is called *indistinguishability obfuscation* ($i\mathcal{O}$). Importantly, there is no proof suggesting that $i\mathcal{O}$ is impossible. Nevertheless, this notion received much less attention, since it seemed too weak to be useful. This view changed somewhat when Goldwasser

The original version of this paper is entitled “Candidate Indistinguishability Obfuscation and Functional Encryption for All Circuits” and was published in FOCS 2013.

and Rothblum proved a strong relative guarantee—namely, that an indistinguishability obfuscator is in some sense a *best-possible obfuscator*: it is as good as any other obfuscator of roughly the same complexity.¹⁹ But still, in the absence of intuitive security guarantees and without a plausibly secure construction of iO for general programs, there was not much interest in seeing what could be built from iO .

1.2. Indistinguishability obfuscation: a candidate construction

This article reviews the first plausibly secure construction of iO for general programs.¹¹ Since this proposal in 2013, interest in iO has exploded, and researchers have found that iO is actually very powerful. Surprisingly, it enables many of the “pie-in-the-sky” applications that motivated research on VBB obfuscation.

Our construction has two parts. First, we show how any method for iO that works for “simple programs”—in particular, for log-depth Boolean circuits, denoted by the complexity class NC^1 —can be used to get iO for general programs (modeled as Boolean circuits). Second, we give our “core obfuscator,” namely our iO construction for NC^1 circuits. The bootstrapping transformation from NC^1 circuits to general programs is “cryptographically kosher,” in that we rigorously *prove* that our iO construction for general programs is secure assuming that the iO construction for NC^1 circuits is secure, and assuming cryptographic tools that by now have become widely accepted, such as cryptographic proofs with certain properties and fully homomorphic encryption (FHE)^{13, 14, 23} with a decryption algorithm in NC^1 .

The second part, our core iO construction for NC^1 circuits, is less cryptographically kosher, and this is the reason we call our iO construction a “candidate.” While the security of our NC^1 construction is based on a well-defined mathematical assumption, the assumption is highly non-standard and complicated, having none of the simplicity and elegance of standard cryptographic assumptions like the hardness of factoring large integers. Moreover, our assumption is stated in terms of *cryptographic multilinear maps*,¹⁰ for which constructions have been proposed only recently, and which themselves rely on their own non-standard assumptions. Cryptographers do not intend to tolerate this situation very long, and currently are working furiously and simultaneously on trying to break these assumptions and to base iO on more natural assumptions.

Another caveat is that our iO construction is quite impractical (though we can expect substantial performance improvements in the future). Our construction is a proof of concept that has helped to transform our understanding of obfuscation, not something that should be deployed any time soon.

How does our approach to general-purpose program obfuscation differ from previous approaches? While many diverse approaches have appeared in the literature, a common thread connecting most previous approaches was the idea of obfuscation by “hiding a needle in a haystack”—that is, by hiding the actually important instructions and conditions within the original program among many extraneous commands and tests.⁶ Unfortunately, computers are excellent

at finding needles in haystacks, and these approaches, while natural, do not work very well.

Our approach is very different. At a high level, our core obfuscator creates a globally randomized representation of the program. Intuitively, this globally randomized representation of the program takes each individual instruction from the original program and converts them into highly unnatural randomized instructions. These instructions appear random and useless to an attacker. In particular, each instruction has a random-looking effect on the entire obfuscated program’s state. However, the global randomization that produces these instructions is designed in such a way that the randomization “cancels out” only and exactly when the entire computation is finished. That is, once every one of the weird instructions have been carried out precisely in the specified order, the randomization within these instructions is eliminated and only the true output of the original computation remains. Thus, in our approach, there is neither a haystack nor a needle: every instruction in the obfuscated version of the program is essential; each one moves the computation forward. However, the randomized mathematical representation of the program ensures that until the very end of the computation, every step looks random and uninteresting.

More specifically, our core obfuscator for simple programs combines cryptographic multilinear maps,^{7, 10} a tool somewhat similar to FHE, with variants of tools introduced by Barrington² and Kilian²¹ that allow NC^1 circuits to be efficiently re-expressed as matrix branching programs, which can be randomized in a strong sense. Then, we bootstrap iO from simple programs to complicated programs by encrypting the entire complicated program using FHE, and then conditionally decrypting the result using a simple program that has been obfuscated using our core obfuscator.

1.3. Hiding secrets in software

Why is iO interesting? Although iO is less versatile and trickier to use than VBB obfuscation, it is powerful enough to let us *hide secrets in software* (in some cases).

First, we should mention that there are limits to what any obfuscation can accomplish. For example, suppose that a program P prints its own code. Then, any obfuscation $O(P)$ of P also prints P . If a program is determined to be “exhibitionist,” there is nothing obfuscation can do to prevent it. Even if a program is not so exhibitionist, it might be *learnable*. Roughly, a function is *learnable* if, given only oracle access to the program P , an adversary can efficiently construct a program P' that has the same input–output behavior as P . (Of course, any program is learnable *inefficiently*, but we only aim to defeat polynomial-time adversaries.) For example, consider a program $P_s(x)$ that outputs 1 if $x < s$, and 0 otherwise. The class of programs $\{P_s\}_s$ is learnable, since a simple binary search would allow an adversary to learn the secret s . Even a black box does not hide P if P is learnable from oracle access. So, we will be interested in obfuscating programs that are *unlearnable*: programs that at least can keep secrets secure against efficient adversaries that are given only oracle access.

Cryptography offers many examples of functions that are unlearnable, based on widely believed assumptions such as the existence of one-way functions or the hardness of the

discrete logarithm problem. For example, given any one-way function, one can construct a *pseudorandom function* (PRF), a function that no adversary can efficiently distinguish from a truly random function. A truly random function is unlearnable, and therefore so is a PRF, since otherwise it could be distinguished from a truly random function. Assuming the discrete logarithm problem is hard, one can construct an encryption scheme that is secure against adaptive chosen-ciphertext attack (CCA-secure encryption). An encryption scheme is CCA-secure if the adversary cannot guess anything about what a new ciphertext c encrypts, even after it asks for and obtains the decryptions of other ciphertexts $c_1, \dots, c_i$ of its choice. For a CCA-secure encryption scheme, the decryption function is unlearnable.

The unlearnable functions above have some *secret* inside them—a PRF key or a decryption key. If we had black-box obfuscation, we could certainly prevent this secret from being extracted; we could hide secrets in software.

Of course, we do not have general black box obfuscation. We need to figure out how to hide secrets in software without black boxes, using just $i\mathcal{O}$. We start to address this issue below.

Shell Games with Secrets. Here is a typical “shell game” we play when we want to prove that $i\mathcal{O}$ obfuscation hides a secret. Suppose that (sk_0, pk_0) and (sk_1, pk_1) are independent secret/public key pairs—for example, for a public-key encryption scheme. Suppose that we have two programs P_0 and P_1 that have the same input-output behavior, but P_0 only uses sk_0 (but not sk_1) and P_1 only uses sk_1 (but not sk_0). The definition of $i\mathcal{O}$ guarantees that the obfuscation of P_0 would be indistinguishable from an obfuscation of P_1 , but can we leverage this to show that sk_0 remains hidden?

Indeed we can: Clearly, the obfuscation of P_0 reveals nothing about sk_1 , since P_0 does not know anything about the secret sk_1 . Similarly, an $i\mathcal{O}$ -obfuscation of P_1 would reveal nothing about sk_0 . However, by the security of $i\mathcal{O}$, since P_0 and P_1 are functionally equivalent, it is hard for an adversary to distinguish whether P_0 or P_1 was obfuscated. Therefore, since P_1 *could* have been obfuscated (as far as the adversary knows), the $i\mathcal{O}$ -obfuscation of P_0 also reveals nothing about sk_0 . In other words, since the output of the programs is the same regardless of whether sk_0 or sk_1 is inside, we can use $i\mathcal{O}$ to play a “shell game” to switch the secret inside, and thus argue that neither secret is revealed. Similar “two-key” techniques have been used many times in cryptography, for example to construct CCA-secure encryption schemes.^{22, 24}

As an example of how we use this shell game, take our method of bootstrapping $i\mathcal{O}$ for NC^1 circuits to $i\mathcal{O}$ for general programs (modeled as boolean circuits).

To construct $i\mathcal{O}$ for general programs, our natural starting point is *fully homomorphic encryption* (FHE).^{13, 14, 23} FHE is a special kind of encryption that allows anyone (e.g., the “cloud”) to process or compute on encrypted data while it remains encrypted, without decrypting it. In other words, FHE-encrypted data allows *anyone* to compute an encryption of *any function* of that data. Interpreting data as a program, FHE allows the creation of encrypted programs. An encrypted program, however, is not the same thing as an obfuscated program, since FHE-encrypting a program changes its functionality:

given x as input, the FHE-encrypted program’s output is not $P(x)$, but an *encryption* of $P(x)$. However, FHE still seems like a good place to start, since we already have FHE schemes capable of evaluating general programs (modeled as boolean or arithmetic circuits),¹³ even schemes based on simple well-studied assumptions.^{4, 5, 16} To get obfuscation we need to figure out how to securely decrypt the output of the FHE-encrypted program. But for this to work, we also need to make sure that the user has properly applied FHE to produce an encryption of $P(x)$. Indeed, this is critical, since giving away unrestricted access to the FHE decryption function would give away the farm!

For this *conditional decryption* step, we use our shell game. The obfuscation of a boolean circuit P_0 consists of an FHE encryption of P_0 under *two* different independent secret/public FHE key pairs (sk_0, pk_0) and (sk_1, pk_1) , and a conditional decryption circuit CDec (to be described momentarily) that is obfuscated using our $i\mathcal{O}$ -obfuscator for NC^1 circuits. (To hide the topology of P_0 , we use a universal circuit U such that $U(P_0, x) = P_0(x)$ and encrypt the input P_0 to U .) The evaluator runs both encrypted versions of P_0 on its input, and feeds the resulting ciphertexts, along with a *proof* that they were computed correctly, as input to $i\mathcal{O}(\text{CDec})$, which decrypts the first ciphertext using sk_0 if the proof verifies, and outputs $\perp$ otherwise. The proof here is straightforward: it consists of the input and a step-by-step transcript of the FHE evaluation of the encrypted program on that input. Fortunately, using existing constructions, both the proof verification and FHE decryption can be computed using a NC^1 circuit, as required.

We now need to argue that our construction really does yield a way of bootstrapping $i\mathcal{O}$ for NC^1 to $i\mathcal{O}$ for general programs. Suppose P_1 is a program with the same input-output behavior as P_0 . We transform the obfuscation of P_0 into an obfuscation of P_1 via a sequence of steps. First, we replace the encryption of P_0 under pk_1 with an encryption of P_1 under pk_1 ; this change is indistinguishable to an adversary since sk_1 is never used, and therefore encryptions of any two strings under pk_1 must be indistinguishable from each other. Next, and this is the crucial step, we change CDec so that it contains only sk_1 (not sk_0) and decrypts the second FHE ciphertext using sk_1 (instead of the first ciphertext using sk_0) after verifying the proof. The key point is that input-output behaviors of the old and new versions of CDec are the same, since (by the proof and the equivalence of P_0 and P_1) if the output of CDec is not $\perp$, then the two FHE ciphertexts *must* have the same decryption. Consequently, by the security of $i\mathcal{O}$, this change is indistinguishable to the adversary. We can then proceed in two similar steps, changing the other encryption of P_0 to P_1 and switching the secret key back to sk_0 , to arrive at an obfuscation of P_1 without the adversary being any wiser.

Punctured Programs. Another way to use $i\mathcal{O}$ to hide secrets in software is the punctured program technique of Sahai and Waters.²⁶ We illustrate the technique in Section 4.2, where it is used to convert secret key encryption into public key encryption by obfuscating the encryption function.

1.4. Not quite obfutopia

While we have managed to squeeze useful security guarantees out of $i\mathcal{O}$ (with more positive results based on $i\mathcal{O}$ appearing

regularly in the cryptographic literature), nevertheless cryptographers remain somewhat unsatisfied. One problem is that we do not understand what absolute guarantees we can make about obfuscation itself, or even what sort of security current $i\mathcal{O}$ constructions may give us. The “right” definition for program obfuscation is far from resolved, and researchers have begun to investigate stronger notions—such as differing inputs obfuscation¹—that also seem to escape the impossibility of VBB.

Also, while VBB obfuscation is impossible in general, it remains possible that a broad class of programs, including many “real-world” programs, can be VBB obfuscated. But what is this class of programs?

The reason that VBB obfuscation remains appealing as a notion, despite its problems, is that we have a long list of glorious applications we could build in a world—call it “Obfutopia”—where general VBB program obfuscation is somehow possible. For example, a classic motivation for obfuscation was to build secure autonomous mobile agents that perform transactions on your behalf to your specifications using your secret signing key. The agent may run on untrusted platforms that could try to “torture” it to extract your secrets. While VBB obfuscation would solve this problem, trying to prove that this is secure with a weaker notion of obfuscation is a challenge.

As a more speculative application, suppose that in the distant future we want to upload our minds to cyberspace. Assuming this happens, we might as well do it securely. Can a mind programmed in software have secrets, innermost thoughts? With $i\mathcal{O}$, we can provide some assurance: if a mind can compartmentalize its secrets so that they do not affect its behavior, the secret can remain hidden. But although digital minds presumably would have impressive powers of compartmentalization, it would be more convenient if obfuscation’s assurances were less restrictive.

Like all utopias, Obfutopia has echoes of dystopia. Just as access to secure encryption makes it easier for terrorists to hide their plans, obfuscation can be used for nefarious purposes—for example, by malware designers to hide malicious content. Privacy-enhancing technologies often come with a cost. However, given the ever-increasing list of positive applications of secure obfuscation, we argue that the potential benefits outweigh the potential costs.

2. OBFUSCATION BASICS

Barak et al.¹ formally define an obfuscator as a probabilistic (randomized) algorithm O that takes a program P as input and meets the following three conditions:

1. Functional equivalence: For every program P , the string $O(P)$ describes a program with the same input-output behavior as P .
2. Polynomial slowdown: The description length and running time of $O(P)$ are polynomially related to P .
3. Unintelligibility.

We will define the security of obfuscation (“unintelligibility”) in Section 2.2. For now, let us focus on obfuscation’s functional aspects.

2.1. What is a program?

To make the definition of program obfuscation meaningful, we first need to define what we mean by a “program.” Obviously, we do not want a definition of programs that is language-dependent, so we use theoretical models of computation. In this article, we will use Boolean circuits as our model of computation.

A Boolean circuit is a composition of Boolean gates, such as AND, OR, and NOT gates. The gates are typically arranged into levels, so that the outputs of gates at level i are inputs to gates at level $i + 1$ unless i is the last level of the circuit. A circuit does not contain any loops; it is a directed acyclic graph. The number of gates is called the size of the circuit, and the number of levels is called the depth. For a circuit C , we denote by $|C|$ the size of C .

In a circuit, all loops are unrolled and all branches are evaluated. Unlike a RAM program, which may touch only input that is relevant for a particular execution, a circuit ingests all the input at its input level. Consequently, translating some RAM programs into circuits may add a lot (but still polynomial amount) of overhead. We will ignore this issue here, as efficiency (beyond achieving polynomial overhead) is not our concern here.

An NC^1 family of circuits is one where the depth of each circuit in the family is at most logarithmic in its size.

With these preliminaries, we can make precise the functionality and polynomial slowdown requirements for obfuscation. Namely,

1. Functional equivalence: For every Boolean circuit C , $O(C)$ describes a circuit C' such that for all inputs x , we have $C(x) = C'(x)$.
2. Polynomial slowdown: There exists a polynomial p such that for all Boolean circuits C , the size of the circuit described by $O(C)$ is at most $p(|C|)$.

2.2. What does “unintelligible” mean?

As mentioned in the Introduction, we take *indistinguishability obfuscation* ($i\mathcal{O}$)¹ as our notion of unintelligibility. In the formal definition of $i\mathcal{O}$, security is modeled as a game between a challenger and an adversary. The adversary picks two circuits P_0 and P_1 that are the same size (padding one of the circuits if necessary) and that have the same input-output behavior but which may differ internally. It gives these programs to the challenger. The challenger obfuscates one of them, producing an obfuscation $O(P_b)$ for random $b \in \{0, 1\}$, which it gives to the adversary. Finally, the adversary tries to guess b . The obfuscation is considered secure if no efficient adversary can win this game with probability non-negligibly greater than $1/2$.

In principle, one way to $i\mathcal{O}$ -obfuscate would be to *canonicalize* P_b by outputting the lexicographically first program with the same input-output behavior as P_b , in which case $O(P_0) = O(P_1)$ and the adversary *cannot* win with probability greater than $1/2$. But true canonicalization is inefficient in general unless $P = \text{NP}$, so we must settle for pseudo-canonicalization and require that the distributions of $O(P_0)$ and $O(P_1)$ are computationally indistinguishable to efficient adversaries, in the sense described above: no efficient adversary should

be able to guess the bit b with probability non-negligibly greater than $1/2$.

As mentioned in the Introduction, the work of Goldwasser and Rothblum¹⁹ showed that an (efficient) indistinguishability obfuscator is a “best-possible” obfuscation: it is as good as any other obfuscation of roughly the same complexity.¹⁹ The proof is simple: Suppose $BO(\cdot)$ is the actual best obfuscator of a certain class of programs (modeled as boolean circuits), while $Pad(\cdot)$ merely increases the size of the circuits the same amount as $BO(\cdot)$. Then, for any circuit C , the circuits $BO(C)$ and $Pad(C)$ are the same size and functionality equivalent. Therefore, if iO is an indistinguishability obfuscator, $iO(BO(C))$ and $iO(Pad(C))$ are indistinguishable to efficient adversaries. Since they are indistinguishable, $iO(Pad(C))$ obfuscates C as well as $iO(BO(C))$, which obfuscates C as well as $BO(C)$. Unfortunately, however, “best possible” is only a relative guarantee that does not directly imply any secrets in the obfuscated program remain hidden.

Another justification for iO is that it turns out to be surprisingly useful. In Section 4, we add to the applications of iO mentioned in the Introduction.

3. OUR CORE OBFUSCATOR

Recall that our obfuscation scheme¹¹ works in two steps. In the Introduction, we already sketched how to bootstrap obfuscation for NC^1 circuits to obfuscation for general circuits. Now we will sketch how our “core” obfuscator obfuscates NC^1 circuits. Our core obfuscator builds on a new cryptographic tool called multilinear maps,^{7,10,15} which we review below.

3.1. Cryptographic multilinear maps

As a first approximation, one can say that a cryptographic multilinear map system encodes a value $x \in \mathbb{Z}_p$ (where p is some large prime) by encrypting it using a FHE scheme. The multilinear map system inherits the homomorphism of FHE: given encodings of x and y , one can compute encodings of $x + y$ and $x \cdot y$. But the multilinear map system comes with additional constraints, and additional functionality. Each encoding is associated to some level $i \leq k$ for parameter k . Encodings can only be added if they are at the same level: $Enc_i(x) \oplus Enc_i(y) \rightarrow Enc_i(x + y)$. Encodings can be multiplied, subject to restrictions, and the levels are added: $Enc_i(x) \otimes Enc_j(y) \rightarrow Enc_{i+j}(x \cdot y)$ if $i + j \leq k$ but is meaningless when $i + j > k$. Unlike FHE, the multilinear map system comes equipped with a *zero test*: the system gives an efficient procedure to test whether an encoding at level k encodes the value 0.

Thus, unlike an encryption scheme, the multilinear map system leaks some information about what is encoded (whether the value is zero). Roughly, we carefully construct our core obfuscator so that the leakage of the multilinear map system is positioned precisely to become the leakage of the overall obfuscated program output.

3.2. Our core obfuscator for NC^1 circuits

We now sketch how our “core” obfuscation scheme for NC^1 circuits (circuits of logarithmic depth) works. We confess that many details will be lacking in this short article; see Garg et al.¹¹ for more details. Instead, we will try to give you a glimpse into the main insights that make our approach

work, and a very rough outline of how these insights can be extended to yield an actual obfuscation.

Barrington: A more friendly mathematical representation of NC^1 circuits. Boolean circuits are already an unrealistic representation of general computer programs. Few of us would write code as a Boolean circuit. But to get where we want to go, we move to a “programming language” that is even further removed from reality, but that does not lose any generality. We are not doing this for kicks, though. Our objective is to represent our NC^1 circuit in a way that plays nicely with our cryptographic multilinear maps and allows strong randomization.

The specific programming language that we choose is a variant of a branching program considered by Barrington,² that we call a Matrix Branching Program. In a matrix branching program, the program is described by a number n , and a sequence of pairs of full-rank square matrices

$$\mathcal{M} = (M_{1,0}, M_{1,1}), (M_{2,0}, M_{2,1}), \dots, (M_{k,0}, M_{k,1})$$

The number n , which must be at most k , describes the length of inputs that the program can take as input. Given an input $x \in \{0, 1\}^n$, to “run” the program $\mathcal{M}$ on x , we compute the matrix product:

$$\prod_{i \in \{1, \dots, k\}} M_{i, x[i \bmod n]}$$

where here $x[j]$ denotes the j th bit of x , where j can range from 0 to $n - 1$.

A matrix branching program is valid if this product always yields either the identity matrix I , or some other fixed matrix $B \neq I$. If the product is I , then we say that the output of the program $\mathcal{M}(x)$ is 1. If the product is B , then we say that the output of the program $\mathcal{M}(x)$ is 0.

This representation is useful because Barrington showed that any NC^1 circuit can be efficiently re-expressed as a polynomial-size matrix branching program. Also, matrix product is a multilinear operation: every entry in each matrix occurs exactly once in every monomial of the product, so that we can use the multilinear map system to encode every entry of each matrix, and compute the product correctly using the multilinear map operations described above. Lastly, as we describe next, matrix branching programs can be *globally randomized* in a strong sense.

Kilian: Global Randomization of Branching Programs. The entire global randomization that we actually use to obtain our obfuscation, and why it works, is frankly too complex to describe at length here. Instead, we will ask you to suspend disbelief as we describe how to obfuscate a special case of programs: namely, programs that have no input at all. Now, the thoughtful reader will complain that this special case is utterly trivial to obfuscate, and indeed it is: One could simply pre-compute the output of the program, and provide this output as the obfuscation. This would absolutely be a secure obfuscation—although this is not what we are going to do. We ask you to put this thought aside. We will sketch a different way to deal with this special case, and we assure you that our method for dealing with this special case will provide insight into *why* our actual global randomization works in the general case.

What does a matrix branching program with no input look like? Instead of a sequence of pairs of matrices, it would just be a sequence of matrices:

$$\mathcal{M} = M_1, M_2, \dots, M_k$$

and its output would be the product

$$\prod_{i \in \{1, \dots, k\}} M_i$$

How can we globally randomize these matrices, so that the output is preserved, but the “steps” of the computation reveal nothing more than the output? The answer is to apply a technique of Kilian,²¹ as follows: Choose completely random invertible matrices $R_1, \dots, R_{k-1}$ over $\mathbb{Z}_p$, and set $R_0 = R_k = I$. Now, define

$$\tilde{M}_i = R_{i-1} M_i R_i^{-1}$$

Then, we can observe that

$$\prod_{i \in \{1, \dots, k\}} M_i = \prod_{i \in \{1, \dots, k\}} \tilde{M}_i$$

However, a simple inductive argument shows that the entire collection of matrices $\tilde{M}_i$ contain *no information* beyond the product $\prod_i M_i$. For example, suppose $k = 2$. In other words, we only have two matrices $\tilde{M}_1 = M_1 R^{-1}$ and $\tilde{M}_2 = R M_2$. Then, by letting $Q = M_2^{-1} R^{-1}$ and algebraic rearrangement, we see that equivalently, we could set $\tilde{M}_1 = M_1 M_2 Q$ and $\tilde{M}_2 = Q^{-1}$. Note that if R is a random invertible matrix, then so is Q . Thus, $\tilde{M}_1$ and $\tilde{M}_2$ can be created knowing only the product $M_1 M_2$, and nothing else about the original “steps” of the computation M_1 and M_2 . This proof generalizes to any k .

This insight shows that it is in fact possible to globally randomize a program, using only local randomizations with a set of correlated random variables, in a way that preserves the output, but completely randomizes the individual steps.

Our actual core obfuscator—which must deal with an exponential number of potential inputs—begins with this idea. In essence, this idea is used to argue security on an input-by-input basis. Additional global randomizations are invoked to thwart other algebraic attacks that can arise when inputs are present (e.g., input-mixing attacks, where part of the computation is done with respect to one input, and another part with respect to another input). But the main idea remains the same: to globally randomize the program in a way that preserves the output, but hides all other useful information about the original individual steps of the program and how they combined with each other.

4. APPLICATIONS

Indistinguishability obfuscation has enabled a wide variety of exciting applications that has become so vast that it is impossible to do them justice here. Here, we cover just a few applications to help the reader appreciate the possibilities.

4.1. Witness encryption

Traditionally, encryption is a tool for conveying confidential messages to specific recipients that have secret keys.

Departing from tradition, suppose we would like to encrypt a message not to a fixed recipient who has a secret key, but to anyone who knows a proof for the Riemann Hypothesis (or, more generally, a witness for some NP statement). We want any recipient that has any proof (of some bounded size) of the RH to be able to decrypt the message. We also want to generate this encryption without knowing ourselves whether the RH is true or not. Regarding security, we want that if the RH is false, then the ciphertext should hide the encrypted message. This problem of *witness encryption*¹² (WE) was first posed by Rudich in 1987.

More formally, given an NP Language L , a WE scheme for L is an encryption scheme that takes as input an instance x and a message bit b , and outputs a ciphertext c . If $x \in L$ and w is a valid witness for x , then a decryptor can use w to decrypt c and recover b . However, if $x \notin L$, then an encryption of 0 should be computationally indistinguishable from an encryption of 1.

Using $i\mathcal{O}$, we can easily construct a WE scheme for an NP-Complete language such as SATISFIABILITY. Specifically, consider the circuit $C_{x,b}(w)$ (with x and b hard-coded in it) defined as follows: on input w , $C_{x,b}$ outputs b if w is a valid witness for $x \in L$, and otherwise just outputs $\perp$ (indicating “undefined”). To encrypt b , the encrypter simply outputs an $i\mathcal{O}$ obfuscation of $C_{x,b}$. With a legitimate witness w for $x \in L$, anyone can evaluate the obfuscated circuit to recover b . On the other hand, if $x \notin L$, then both $C_{x,0}$ and $C_{x,1}$ are constant functions that always output $\perp$, and $i\mathcal{O}$ guarantees that the obfuscation of these equivalent functions are computationally indistinguishable.

4.2. Punctured programming: secret-key to public-key encryption

Diffie and Hellman⁸ already described some of the potential of obfuscation back in 1976 in their “New Directions” paper. They noticed that obfuscation could potentially be used to transform a secret key encryption scheme (where sender and receiver use the same secret key) into a public key encryption (PKE) scheme (where the sender uses a “public” key for encryption and only the receiver knows the secret key for decryption). Specifically, take the encryption procedure or program, with the secret key embedded inside it, and publish its obfuscation as the public key. Senders could use this public obfuscated program to encrypt messages, but (hopefully) not to decrypt ciphertexts, as the secret key remains hidden inside the obfuscated program.

Given ideal black box obfuscation, this technique works well for some secret key encryption schemes. (Beware that it fails completely for some others, like stream ciphers, where encryption is an involution.) Can we make it work for $i\mathcal{O}$? Indeed, we can: in fact, we have two quite different ways to construct public-key encryption from $i\mathcal{O}$ and any one-way function (OWF).

The first technique combines WE with a length-doubling pseudorandom generator (PRG) G . Such a PRG can be built from any OWF, and has the property that $G(x) \in \{0, 1\}^{2n}$ is computationally indistinguishable from a random string when $x \in \{0, 1\}^n$ is sampled uniformly. In the PKE scheme, the secret key is a uniform $w \in \{0, 1\}^n$, and the public key is

$W = G(w)$. To encrypt a bit b , the sender generates a witness encryption of b that can be decrypted by any “witness” w' such that $W = G(w')$. The legitimate decrypter can decrypt using w . Security follows from three observations: (1) By the security of the PRG, an adversary cannot distinguish whether W is a uniform $2n$ -bit string, and (2) If W were a uniform $2n$ -bit string, then (with high probability) it is not in the image of the PRG, and thus there is no valid witness for W , and (3) If there is no valid witness, the security of WE guarantees that b remains hidden.

Sahai and Waters²⁶ found a way of building PKE from iO that is quite close to what Diffie and Hellman had in mind. To do so, they introduced a general technique called *punctured programming*, that has turned out to have many applications. The high-level idea is “to alter a program (which is to be obfuscated) by surgically removing a key element of the program, without which the adversary cannot win the security game it must play, but in a way that does not alter the functionality of the program.”²⁶ This is perhaps best explained by jumping into the technical details. The PKE scheme uses a length-doubling pseudorandom generator (PRG) G , as described above. The scheme also uses a pseudorandom function (PRF) F , which takes a key and a $2n$ -bit string, and outputs a bit that is computationally indistinguishable from uniform. However, we want our PRF to have a special property: we want it to be “puncturable.” A *puncturable PRF* has the property that for any key K and input $\tilde{r}$, there is a compact punctured key $K_{\tilde{r}}$ that allows one to evaluate $F(K, x)$ for any $x \neq \tilde{r}$, but keeps the value of $F(K, \tilde{r})$ pseudorandom. (Puncturable PRFs can also be built from any OWF.) Now, consider a randomized secret key encryption scheme, where the secret key is a key K for the PRF, one encrypts m with randomness r as $Enc_K(r, m) = (G(r), F(K, G(r)) \oplus m)$, and the ciphertext is decrypted in the obvious way using K . Sahai and Waters show that they can use iO to obfuscate the encryption function while leaving decryption unchanged. In this PKE scheme, decryption amounts to evaluating the (puncturable) PRF. No other PKE scheme has such a fast decryption algorithm!

To prove security, Sahai and Waters use their *punctured program* technique. For any $\tilde{r}$, we define a punctured encryption $Enc_{K_{\tilde{r}}}(r, m) = (G(r), F(K_{\tilde{r}}, G(r)) \oplus m)$, which uses the punctured key $K_{\tilde{r}}$ to evaluate the PRF except at $\tilde{r}$, where it outputs $\perp$ (indicating “undefined”). Sahai and Waters observe that the encryption function will never evaluate the PRF on any $\tilde{r}$ outside the range of the PRG. Therefore, for any such $\tilde{r}$, the original encryption function has the same functionality as $Enc_{K_{\tilde{r}}}$, and iO will make them indistinguishable. Now, by the security of the PRG, the adversary cannot distinguish a legitimate ciphertext from a fake ciphertext $(\tilde{r}, F(K, \tilde{r}) \oplus m)$ where $\tilde{r}$ is chosen uniformly, and is therefore with high probability outside the range of the PRG. But since the punctured public key (the obfuscation of $Enc_{K_{\tilde{r}}}$) contains no usable information about $F(K, \tilde{r})$ (even the secret key $K_{\tilde{r}}$ inside provides no help), m appears pseudorandom.

4.3. Functional encryption

Consider a hospital administrator who has access to sensitive patient medical information—specifically medical history and gene sequences of the patients. In the same

hospital a medical researcher is working on establishing a link between a specific set of genes and a type of cancer. In order to validate his findings the researcher needs access to the medical records of the patients in the hospital. The hospital administrator would like to help the researcher but is legally obliged to keep patient records private. In such scenarios, patient data is typically collected before the queries that the medical researcher might be interested in are known. Functional encryption (FE)^{3, 25} enables an elegant solution to this problem. The administrator could encrypt the data as it is created and later provide a specialized secret key to the medical researcher allowing him to learn only the output of his analysis algorithm on the patient data, and nothing more.

More formally, in FE, ciphertexts encrypt inputs x and keys are issued for functions f . The striking feature of this system is that using the key SK_f to decrypt a ciphertext $Enc(x)$, yields the value $f(x)$ but does not reveal anything else about x . Furthermore, *any arbitrary* collusion of key holders relative to many functions f_i does not yield any more information about x beyond what is “naturally revealed” by the functionality (i.e., $f_i(x)$ for all i). More precisely, the security definition of FE requires that the adversary cannot distinguish between encryptions of x_0 and x_1 , even when it is provided with many secret keys SK_{f_i} for which $f(x_0) = f(x_1)$. Note that keys not satisfying this constraint allow the adversary to trivially distinguish between encryptions of x_0 and x_1 .

We next sketch our construction of a FE for all circuits¹¹ from iO . In order to build intuition we start with a solution assuming ideal black box obfuscation. We start by selecting (PK, SK) , a public-secret key pair for a regular public-key encryption scheme. An encryption of a value x under the FE scheme will simply be an encryption of x using the public key PK . The secret key SK_f corresponding to a circuit f is an obfuscation of a circuit that takes a ciphertext c as input, uses SK (hardcoded inside the obfuscated circuit) to decrypt c obtaining x , and then outputs $f(x)$. Intuitively the obfuscation will hide the secret key SK , achieving the desired security.

To adapt the above construction to iO , we use the two-key shell game described in the Introduction. In particular, we set the public key for our FE scheme to be two public keys PK_0 and PK_1 . An encryption of x consists of encryptions of x under both keys, together with a special zero knowledge proof that the two encryptions encrypt the same value. The secret key SK_f is an obfuscated program that checks the proof, uses one of the secret keys to obtain x , and then outputs $f(x)$.

As in the shell game in the Introduction, if the proof is true, then the output of SK_f is the same whether it uses SK_0 or SK_1 , and the strategy is to use iO to argue that these cases are indistinguishable. However, ultimately we need to argue that the adversary cannot distinguish an encryption of x_0 from an encryption of x_1 , and for this our proof needs to use an intermediate hybrid experiment where a “fake” simulated cryptographic proof is generated, and the two ciphertexts are actually encryptions of x_0 and x_1 respectively. But here we encounter a new problem: if it is possible to generate “fake” cryptographic proofs that “prove” the ciphertexts

encrypt the same value when they do not, then we cannot use iO to transition from SK_0 to SK_1 as described above.

We address this problem by introducing the notion of *statistically simulation sound* cryptographic proofs, which require that valid proofs only exist for true statements, with *exactly one* exception: there is a single false statement Ψ^* such that there exists a valid proof π^* for Ψ^* . However, for every other statement Ψ , if a valid proof π exists for Ψ , then Ψ must be true. We show how to achieve statistical simulation soundness using witness-indistinguishable proofs.⁹ Once we have it, then a similar two-key alternation argument to the one given in the Introduction allows us to establish security of our FE scheme. We note that full collusion-resistance follows from a standard hybrid argument using the iO definition for each secret key given out, one at a time.

4.4. Restricted-use software

Here we discuss an application of iO to a question that does not deal with encryption or authentication, but rather *restricted-use software*.

Software developers will often want to release a demo or restricted use version of their software that limits the features that are available in a full version. In some cases a commercial software developer will do this to demonstrate their product; in other cases the developer will want to make multiple tiers of a product with different price points. In other domains, the software might be given to a partner that is only partially trusted and the developer only wants to release the features needed for the task.

Ideally, a developer could create a downgraded version of software simply by starting with the full version and then turning off certain features at the interface level—requiring minimal additional effort. However, if this is all that is done, it could be easy for an attacker to bypass these controls and gain access to the full version or the code behind it. The other alternative is for a software development team to carefully excise all unused functionality from the core of the software. Removing functionality can become a very time consuming task that could itself lead to the introduction of software bugs. In addition, in many applications it might be unclear what code can and cannot remain for a restricted use version. For instance, a subroutine could be written in a highly general form, where only a subset of its functionality is needed for several applications. In this case, the subroutine would have to be re-designed to “cripple” it, a time-consuming and potentially difficult task.

iO provides an immediate solution: the developer can restrict the use at the interface level and then release an obfuscated version of the program. For this application iO suffices, since by definition a version restricted in the interface is indistinguishable from an obfuscated program with equivalent behavior that has its parts removed or redesigned at the start.

5. CONCLUSION

We described a candidate indistinguishability obfuscation scheme, and some of the exciting applications it makes possible. However, a lot remains to be done—to base obfuscation on well-established assumptions, to build a practical scheme,

to truly understand what obfuscation means—before cryptographic obfuscation can move from being a proof of concept to a widely-deployed privacy enhancing technology. **□**

References

- Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K. On the (im)possibility of obfuscating programs. In *CRYPTO* (2001). Springer, 1–18.
- Barrington, D.A. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1 . In *STOC* (1986). ACM, 1–5.
- Boneh, D., Sahai, A., Waters, B. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference (TCC)* (2011). Springer, 253–273.
- Brakerski, Z., Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) lwe. *SIAM J. Comput.* 43, 2 (2014), 831–871.
- Brakerski, Z., Vaikuntanathan, V. Lattice-based FHE as secure as PKE. In *Innovations in Theoretical Computer Science (ITCS)* (2014). ACM, 1–12.
- Collberg, C., Nagra, J. *Surreptitious Software: Obfuscation, Watermarking, and Tamperproofing for Software Protection*, 1st ed. Addison-Wesley Professional, Boston, MA, 2009.
- Coron, J.-S., Lepoint, T., Tibouchi, M. Practical multilinear maps over the integers. In *CRYPTO* (2013). Springer, 476–493.
- Diffie, W., Hellman, M.E. New directions in cryptography. *IEEE Trans. Inform. Theory* 22, 6 (1976), 644–654.
- Feige, U., Shamir, A. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO '89* (1990). Springer, 526–544.
- Garg, S., Gentry, C., Halevi, S. Candidate multilinear maps from ideal lattices. In *EUROCRYPT* (2013). Springer, 1–17.
- Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS* (2013). IEEE, 40–49.
- Garg, S., Gentry, C., Sahai, A., Waters, B. Witness encryption and its applications. In *STOC* (2013). ACM, 467–476.
- Gentry, C. A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009). crypto.stanford.edu/craig.
- Gentry, C. Computing arbitrary functions of encrypted data. *Commun. ACM* 53, 3 (2010), 97–105.
- Gentry, C., Gorbunov, S., Halevi, S. Graph-induced multilinear maps from lattices. In *Theory of Cryptography Conference (TCC)* (2015). Springer, 498–527.
- Gentry, C., Sahai, A., Waters, B. Homomorphic encryption from learning with errors: Conceptually simpler, asymptotically faster, attribute-based. In *CRYPTO* (2013). Springer, 75–92.
- Goldwasser, S., Kalai, Y.T. On the impossibility of obfuscation with auxiliary input. In *FOCS* (2005). IEEE, 553–562.
- Goldwasser, S., Micali, S. Probabilistic encryption. *J. Comput. Syst. Sci.* 28, 2 (1984), 270–299.
- Goldwasser, S., Rothblum, G.N. On best-possible obfuscation. In *Theory of Cryptography Conference (TCC)* (2007). Springer, 194–213.
- Hada, S. Zero-knowledge and code obfuscation. In *ASIACRYPT* (2000). Springer, 443–457.
- Kilian, J. Founding cryptography on oblivious transfer. In *STOC* (1988). ACM, 20–31.
- Naor, M., Yung, M. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC* (1990). ACM, 427–437.
- Rivest, R., Adleman, L., Dertouzos, M.L. On data banks and privacy homomorphisms. In *Foundations of Secure Computation* (1978), 169–180.
- Sahai, A. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS* (1999). IEEE, 543–553.
- Sahai, A., Waters, B. Fuzzy identity-based encryption. In *EUROCRYPT* (2005). Springer, 457–473.
- Sahai, A., Waters, B. How to use indistinguishability obfuscation: Deniable encryption, and more. In *STOC* (2014). ACM, 475–484.

Sanjam Garg (sanjamg@berkeley.edu), Department of EECS, UC Berkeley, Berkeley, CA.

Craig Gentry and Shai Halevi ([[@us.ibm.com](mailto:cbgentry.shaih)], Cryptography Research Group, IBM TJ Watson Research Center, Yorktown Heights, NY.

Mariana Raykova (mariana@cs.columbia.edu), Computer Science Laboratory, SRI International, Menlo Park, CA.

Amit Sahai (sahai@cs.ucla.edu), Department of Computer Science, UCLA, Los Angeles, CA.

Brent Waters (bwaters@cs.utexas.edu), Department of Computer Science, UT Austin, Austin, TX.

Technical Perspective

Software *Is* Natural

By Gail C. Murphy

IN JUST HALF of the average 150-year life span of a Galápagos tortoise, high-level programming languages have come into existence and have undergone a remarkable transformation. The earliest forms of higher-level languages made it easier to state mathematical expressions than was possible in machine code, but it was not long before the need for languages to go beyond mathematical computations and to express solutions for business problems was recognized. One of the languages designed in response to this need was Cobol, with a stated desire to maximize the use of simple English phrases. From this point on, it became more common for programming languages to include a mix of keywords and constructs based on natural language phrases. Combined with the ability for programmers to state variable names using semantically meaningful tokens and the ability to embed comments that explain the code in natural language, programming languages continue to ease the expressions of complex ideas into software.

The ability to more easily express programs has enabled the creation of software to tackle complex problems. This software is often large and complex itself. To understand, extend, and solve problems in this intricate software, software engineers need tools. Many tools have been created, from ones that indicate where the expression of software does not meet syntactic rules of the language to defect predictors that aim to suggest where bugs may be lurking in the software and where effort might be directed to eradicate such bugs before they affect users of the software. The vast majority of the tools built to aid software engineers use deep knowledge of a programming language, often requiring specific work for each kind of programming language and often requiring extensive development effort.

The following work by Hindle et al. takes an entirely new approach to pro-

The following paper takes an entirely new approach to providing tools to help build software.

viding tools to help build software. The authors demonstrate how the phrases used to express software are often repetitive and predictable, similar to how natural languages, like English, are used in practice. Given these similarities, corpus-based statistically rigorous methods that are now being applied to natural language may also be of use when applied to help understand and build software.

In taking this approach, the authors demonstrate some fascinating characteristics of software. For example, the ways in which humans express software are more regular than how humans express text in English. At first glance, this result does not seem so surprising; after all, programming languages, like Java, require programmers to express the desired software in the framework of many keywords and syntactic constructs. However, the authors show this regularity is not the result of the syntax, rather it is due to the project and application domain for which the program is being expressed. The authors show this regularity goes beyond Java to other programming languages as well.

The treatment of software by statistical approaches built for natural language opens up many possibilities for building flexible tools to help develop software. For example, other researchers have now used statistical language

modeling to develop translation tools from an API in one programming language to an API in another programming language, building on existing techniques used by Google for its translation service.

One of the exciting aspects of this paper is the range of potential uses of statistical language models to software. For example, the approach can apply seamlessly to both code and to natural language documents associated with the code, whether they are bug reports, commit messages, design documents, or Web-based question and answer sites. This seamless application enables the use of the approaches to associate a range of documentation with code, enabling tools that suggest code to use given a natural language description or vice versa, to suggest documentation for a given piece of code. Even more provocative, the authors describe how the overall approach might be used to create scalable approximations for determining usually costly program properties, such as a mined rule about a protocol or a fact about how or which storage might be accessed when a program is executed.

It is not often that research is conducted that changes the course of a field. The demonstration by the authors that software is natural and that statistical language models apply fundamentally opens up new approaches to creating scalable, useful software development tools. 

Gail C. Murphy (murphy@science.ubc.ca) is a professor of computer science at the University of British Columbia, Vancouver, BC, Canada.

On the Naturalness of Software

By Abram Hindle, Earl T. Barr, Mark Gabel, Zhendong Su, and Premkumar Devanbu

Abstract

Natural languages like English are rich, complex, and powerful. The highly creative and graceful use of languages like English and Tamil, by masters like Shakespeare and Avvaiyar, can certainly delight and inspire. But in practice, given cognitive constraints and the exigencies of daily life, most human utterances are far simpler and much more repetitive and predictable. In fact, these utterances can be very usefully modeled using modern statistical methods. This fact has led to the phenomenal success of statistical approaches to speech recognition, natural language translation, question-answering, and text mining and comprehension.

We begin with the conjecture that *most software is also natural*, in the sense that it is created by humans at work, with all the attendant constraints and limitations—and thus, like natural language, it is also likely to be repetitive and predictable. We then proceed to ask whether (a) code can be usefully modeled by statistical language models and (b) such models can be leveraged to support software engineers. Using the widely adopted *n*-gram model, we provide empirical evidence supportive of a positive answer to both these questions. We show that code is also very regular, and, in fact, even more so than natural languages. As an example use of the model, we have developed a simple code completion engine for Java that, despite its simplicity, already improves Eclipse's completion capability. We conclude the paper by laying out a vision for future research in this area.

1. INTRODUCTION

Of all that we do, *communicating* is the one action that is arguably the most essentially human. Honed by millions of years of cultural and biological evolution, language and communication are an ordinary, instinctive part of everyday life (see Pinker³⁹). Although “natural” languages such as English and Tamil accommodate exquisitely complex forms of expression, everyday human communication evolved in settings impaired by noise and impelled by need; thus, it is most often simple, expedient, even repetitive. This “naturalness” of quotidian human linguistic behavior, together with large online corpora of utterances, modern computing resources, and statistical innovations, has led to a revolution in *natural*-language processing, whose fruits we enjoy every day in the form of speech recognition in mobile devices and automated language translation in the cloud, *inter alia*.

However, the field of natural language processing (“NLP,” see Sparck-Jones⁴⁶ for a brief history) went through several decades of rather slow and painstaking progress, beginning with early struggles with dictionary and grammar-based efforts in the 1960's. In the 1970s and 1980s, the field was reanimated with ideas from logic and formal semantics, which still proved

too cumbersome to perform practical tasks at scale. Both these approaches essentially dealt with NLP from first principles—addressing *language*, in all its rich theoretical glory, rather than examining corpora of actual *utterances*, that is, what people actually write or say. In the 1980s, a fundamental shift to *corpus-based, statistically rigorous* methods occurred. The availability of large, on-line corpora of natural language text, including “aligned” text with translations in multiple languages,^a along with the computational muscle (CPU speed, primary and secondary storage) to estimate robust statistical models over very large data sets has led to stunning progress and widely available practical applications, such as statistical translation used by translate.google.com.^b

Can we apply these techniques *directly* to software, with its strange syntax, awash with punctuation, and replicate this success? The funny thing about programming is that it is as much *an act of communication*, from one human to another, as it is a way to tell computers what to do. Knuth said as much, 30 years ago:

*Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.*²³

If one, then, were to view programming as an act of communication, is it driven by the “language instinct”? Do we program as we speak? Is our code largely simple, repetitive, and predictable? *Is code natural?*

This, precisely, is our central hypothesis:

Programming languages, in theory, are complex, flexible and powerful, but, “natural” programs, the ones that real people actually write, are mostly simple and rather repetitive; thus they have usefully predictable statistical properties that can be captured in statistical language models and leveraged for software engineering tasks.

We believe that repetitiveness occurs in code corpora at *lexical, syntactic, and semantic* levels, both locally and globally, and we believe that this phenomenon can be captured and exploited using modern methods of computational

^a This included the Canadian Hansard (parliamentary proceedings), and similar outputs from the European parliament.

^b Indeed, a renowned pioneer of the statistical approach, Fred Jelinek, is reputed to have exclaimed: “Every time a linguist leaves our group, the performance of our speech recognition goes up!” See http://en.wikiquote.org/wiki/Fred_Jelinek.

The original version of this paper was published in the *Proceedings of the 34th International Conference on Software Engineering* (2012), 837–847. All five authors were at UC Davis at the time.

data analysis. We believe this is a general, useful and practical notion that, together with the very large publicly available corpora of open-source code, will enable a new, rigorous, statistical approach to a wide range of applications, in program analysis, error checking, software mining, program summarization, and code searching.

This paper is the first step in what we hope will be a long and fruitful journey. We make the following contributions:

1. We provide support for our central hypothesis by instantiating a simple, widely used statistical language model, using modern estimation techniques over large software corpora;
2. We demonstrate, using standard cross-entropy and perplexity measures, that the above model is indeed capturing the high-level statistical regularity that exists in software at the n -gram level (probabilistic chains of tokens);
3. We illustrate the use of such a language model by developing a simple code suggestion tool that already substantially improves upon the existing suggestion facility in the widely-used Eclipse IDE; and
4. We lay out our vision for an ambitious research agenda that exploits large-corpus statistical models of natural software to aid in a range of different software engineering tasks.

2. MOTIVATION & BACKGROUND

There are many ways one could exploit the statistics of natural programs. We begin with a simple motivating example. We present more ambitious possibilities later in the paper.

Consider a speech recognizer, receiving a (noisy) radio broadcast, such as *In Berlin today, Chancellor Angela <radio buzz> announced...* A good speech recognizer might guess that the buzzed-out word was *Merkel* from context. Likewise, consider an integrated development environment (IDE) into which a programmer has typed in the partial statement: `for (i=0; i<10.` In this context, it would be quite reasonable for the IDE to suggest the completion `“; i++”` to the programmer.

Why do these guesses seem so reasonable to us? In the first case, the reason lies in the highly predictable nature of newscasts. News reports, like many other forms of culturally contextualized and stylized natural language expression, tend to be well-structured and repetitive. With a reasonable prior knowledge of this style, it is quite possible to fill in the blanks. Thus, if we hear the word “Chancellor Angela,” we can expect that, *in most cases*, the next word is “Merkel.” This fact is well-known and exploited by speech recognizers, natural language translation devices, and even some optical character recognition (OCR) tools. The second example relies on a lesser-known fact: *natural programs are quite repetitive*. This fact was first observed and reported in a very large-scale study of code by Gabel and Su,¹³ which found that code fragments of surprisingly large size tend to reoccur. Thus, if we see the fragment `for (i=0; i<10` we know what follows in most cases. In general, if we know the most likely sequences in a code body, we can often help programmers complete code. Our intuition essentially amounts to the following: *We*

can use a code corpus to estimate the probability distribution of code fragments. If this distribution has low-entropy, and we can model it well: then, given a partial fragment of code (*qua* partial token sequence, partial AST, or partial semantic abstraction) we can often, with high confidence, guess the rest.

What should the form of such a distribution be, and how should we estimate its parameters? In NLP, these distributions are called “language models.” Language models in NLP can be lexical (token sequences), syntactic (grammatical structures), or semantic models (probabilistic or vector-space representations of meaning). Markovian models of token sequences are the simplest, and we use them as an illustration.

2.1. Language models

A language model assigns a probability to an utterance. For us, “utterances” are programs. More formally, consider a set of allowable program tokens^c $\mathcal{T}$, and the (over-generous) set of possible program sequences $\mathcal{T}^*$; we assume the set of possible implemented systems to be $S \subset \mathcal{T}^*$. A language model is a probability distribution $p(\cdot)$ over systems $s \in S$, that is,

$$\forall s \in S [0 < p(s) < 1] \wedge \sum_{s \in S} p(s) = 1$$

In practice, given a corpus C of programs $C \subseteq S$, and a suitably chosen parametric distribution $p(\cdot)$, we attempt to calculate a maximum-likelihood estimate of the parameters of this distribution; this gives us an estimated language model. A great many language models, based on phenomena at lexical, syntactic and semantic levels have been developed (see the Manning and Schütze³¹ book). The choice of a language model is usually driven by practicalities: how easy is it to estimate and how useful it is. For these reasons, the most ubiquitous is the n -gram model, which we now use to illustrate our thesis.

Illustration: N -gram models Consider $a_1 a_2 \dots a_i \dots a_n$, the sequence of tokens in a document (in our case, source code for a system s). We can statistically model how likely tokens are to follow other tokens. Thus, we can estimate the probability of a document based on the product of a series of conditional probabilities:

$$p(s) = p(a_1)p(a_2 | a_1)p(a_3 | a_1 a_2) \dots p(a_n | a_1 \dots a_{n-1})$$

These n -gram models assume a *Markov property*, that is, token occurrences are influenced only by a limited prefix of length $n-1$, thus for 4-gram models, we assume

$$p(a_i | a_1 \dots a_{i-1}) \simeq p(a_i | a_{i-3} a_{i-2} a_{i-1})$$

These models are estimated from a corpus using simple maximum-likelihood based frequency-counting of token sequences. Thus, if “*” is a wildcard, we ask, how relatively often are the tokens a_1, a_2, a_3 followed by a_4 :

$$p(a_4 | a_1 a_2 a_3) = \frac{\text{count}(a_1 a_2 a_3 a_4)}{\text{count}(a_1 a_2 a_3 *)}$$

^c Here we use a token to mean its lexeme.

In practice, estimation of n -gram models is quite a bit more complicated. The main difficulties arise from data sparsity, that is, the richness of the model in comparison to the available data. For example, with 10^4 token vocabulary, a trigram model must estimate 10^{12} coefficients; but even the biggest systems only have $O(10^8)$ tokens. Thus, some trigrams may never occur in one corpus, but may in fact occur elsewhere. This can cause technical difficulties; when we encounter a previously unseen n -gram, we are, in principle, “infinitely surprised,” because an “infinitely improbable” event x estimated from the previously seen corpus to have $p(x) = 0$ actually occurs. *Smoothing* is a technique to handle cases where we have not seen the n -grams yet and still produce usable results with sufficient statistical rigor. Fortunately, there exist a variety of techniques for smoothing the estimates of a very large number of coefficients, some of which are larger than they should be and others smaller. Sometimes it is better to back off from a trigram model to a bigram model. The technical details are beyond the scope of this paper, but can be found in any advanced NLP textbook. In practice we found that Kneser-Ney smoothing (e.g., Koehn,²⁴ Section 7) gives good results for software corpora. However, we note that these are very early efforts in this area, and new software language models^{35, 49} and estimation techniques have improved on the results presented below.

But how do we know when we have a good language model?

2.2. What makes a good model?

Given a repetitive and highly predictable corpus of documents (or programs), a good model captures the regularities in the corpus. Thus, a good model, estimated carefully from a representative corpus, will predict with good confidence the contents of a new document drawn from the same population. A better model can guess the contents of the new document with higher probability. In other words, the model will find a new document with “typical” content not particularly surprising, or very “perplexing.” In NLP, this idea is captured by a measure called *perplexity*, or its log-transformed version, *cross-entropy*.^d Given a document $s = a_1 \dots a_n$, of length n , and a language model $\mathcal{M}$, we assume that the probability of the document estimated by the model is $p_{\mathcal{M}}(s)$. We can write down the cross-entropy measure as:

$$H_{\mathcal{M}}(s) = -\frac{1}{n} \log p_{\mathcal{M}}(a_1 \dots a_n)$$

and by the formulation presented in Section 2.1:

$$H_{\mathcal{M}}(s) = -\frac{1}{n} \sum_{i=1}^n \log p_{\mathcal{M}}(a_i | a_1 \dots a_{i-1})$$

This is a measure of how “surprised” a model is by the given document. A good model estimates higher probabilities (closer to 1, and thus lower absolute log values) to most words in the document. If one could manage to deploy a (hypothetical) truly superb model within an IDE to help programmers complete code fragments, it might be able to guess, with

high probability, most of the program, so that most of the programming work can be done by just hitting a tab key! In practice of course, we would probably be satisfied with a lot less.

But how good are the models that we can actually build for “natural” software? Is software really as “natural” (i.e., unsurprising) as natural language?

3. METHODOLOGY & FINDINGS

To explore these questions, we performed a series of experiments with both natural language and code corpora, first comparing the “naturalness” (using cross-entropy) of code with English texts, and then comparing various code corpora to each other to gain further insight into the similarities and differences between code corpora.

Our natural language studies were based on two very widely used corpora: the Brown corpus, and the Gutenberg corpus.^e For code, we used several sets of corpora, including a collection of Java projects, as well a collection of applications from Ubuntu, broken up into application domains. All are listed in Table 1.

After removing comments, the C and Java project source code was lexically analyzed to produce token sequences that were used to estimate n -gram language models. While

^e We retrieved these corpora from <http://www.nltk.org/>.

Table 1. The 10 Java Projects, C code from 10 Ubuntu 10.10 Categories, and 3 English Corpora we used in our study.

Java Project	Version	Lines	Tokens	
			Total	Unique
Ant	20110123	254457	919148	27008
Batik	20110118	367293	1384554	30298
Cassandra	20110122	135992	697498	13002
Eclipse-E4	20110426	1543206	6807301	98652
Log4J	20101119	68528	247001	8056
Lucene	20100319	429957	2130349	32676
Maven2	20101118	61622	263831	7637
Maven3	20110122	114527	462397	10839
Xalan-J	20091212	349837	1085022	39383
Xerces	20110111	257572	992623	19542

Ubuntu Domain	Version	Lines	Tokens	
			Total	Unique
Admin	10.10	9092325	41208531	1140555
Doc	10.10	87192	362501	15373
Graphics	10.10	1422514	7453031	188792
Interpreters	10.10	1416361	6388351	201538
Mail	10.10	1049136	4408776	137324
Net	10.10	5012473	20666917	541896
Sound	10.10	1698584	29310969	436377
Tex	10.10	1405674	14342943	375845
Text	10.10	1325700	6291804	155177
Web	10.10	1743376	11361332	216474

English Corpus	Version	Lines	Tokens	
			Total	Unique
Brown	20101101	81851	1161192	56057
Gutenberg	20101101	55578	2621613	51156

English is the concatenation of Brown and Gutenberg. Ubuntu 10.10 Maverick was released on 2010/10/10.

^d http://en.wikipedia.org/wiki/Cross_entropy; see also Ref. 31, Section 2.2, p. 75, Equation (2.50).

our experiments were in C and Java, extending to other languages is trivial, and indeed subsequent work with Python yielded similar results.

The Java projects were our central focus; we used them both for cross-entropy studies and experiments with a language model-based code suggestion plug-in for Eclipse. Table 1 describes the 10 Java projects that we used. *Version* indicates the date of the last revision in the Git repository when we cloned the project. *Lines* is calculated using Unix `wc` on all files within each repository; tokens are extracted from each of these files. *Unique Tokens* refers to the number of distinct tokens that make up the total token count given in the *Tokens* field. For English, the unique token count corresponds to the vocabulary size; for code, this count comprises all the identifiers, keywords and operators.

The Ubuntu application domains were quite large in some cases, ranging up to 9 million lines, 41 million tokens (one million unique). The number of unique tokens is interesting, as it gives a rough indication of the potential “surprisingness” of the project corpus. If these unique token were uniformly distributed throughout the project (which would be very odd indeed), we could expect a cross-entropy of $\log_2(1.15E6)$, or approximately 20 bits. A similar calculation for the Java projects ranges from about 13 bits to about 17 bits.

3.1. Cross-entropy of code

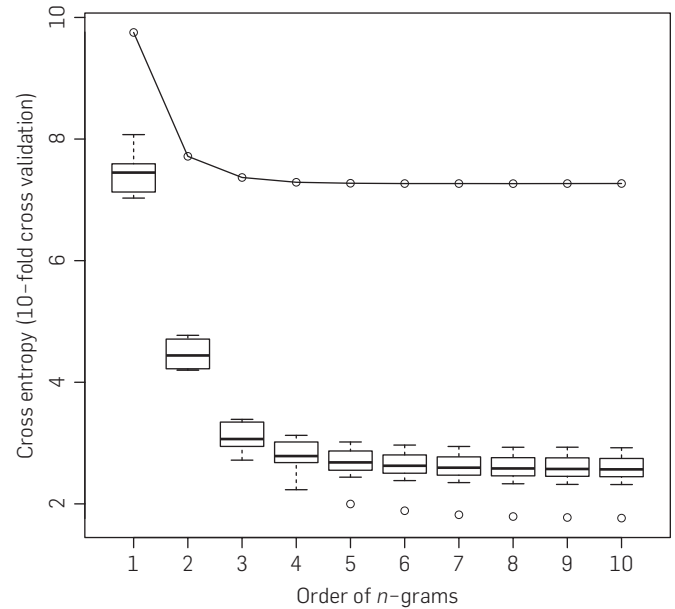
Cross-entropy measures how surprising a test document is to a language model estimated from a corpus. To test a corpus against itself, one must set aside some portion of the corpus for testing, and estimate (train) the model on the rest of the corpus. In all our experiments, we measured cross-entropy by averaging over a 10-fold cross-validation: we split the corpus 90–10% (in lines) at random, trained on the 90%, tested on 10%, and measured the average cross-entropy. A further bit of notation: when we say we measured the cross-entropy of X to Y , Y is the training corpus used to estimate the parameters of the distribution model $\mathcal{M}_Y$ used to calculate the cross-entropy, denoted $H_{\mathcal{M}_Y}(X)$.

First, we wanted to see if there was evidence to support the claim that software was “natural,” in the same way that English is natural, viz., whether regularities in software could be captured by language models.

RQ 1: *Do n -gram language models capture regularities in software?*

To answer this question, we estimated n -gram models for several values of n over both the English corpus and the 10 Java language project corpora, using averages over 10-fold cross validation (each corpus to itself) as described above. The results are in Figure 1. The single line above is the average over the 10 folds for the English corpus, beginning at about 10 bits for unigram models, and trailing down to under 8 bits for 10-gram models. The average self cross-entropy for the 10 projects are shown in boxplots, one for each order from unigram models to 10-gram models. Several observations can be made. First, software unigram entropy is much lower than might be expected from a uniform distribution over unique tokens, because token frequencies are

Figure 1. Comparison of English cross-entropy versus the code cross-entropy of 10 projects.



obviously very skewed.

Second, *cross-entropy declines rapidly with n -gram order*, saturating around tri- or 4-grams. The similarity in the decline in English and the Java projects is striking. This decline suggests that Java corpora, like English corpora, contain a good deal of *locally repetitive text*, and this repetition is *effectively captured by the language model*. We take this to be highly encouraging: the ability to model local repetitions in natural languages has proven to be extremely valuable in statistical NLP; we hope that this regularity can be exploited for software tools (and build an NLP-based code suggestion tool whose success supports the claim).

Last, but not least, *software is far more regular than English* with entropies sinking down to under 2 bits in some cases.

Corpus-based statistical language models can capture a high level of local regularity in software, even more so than in English.

This raises an important question: *Is the increased regularity we are capturing in software merely a difference between English and Java?* Java is certainly a much simpler language than English, with a far more structured syntax. Might not the lower entropy be simply an artifact of the artificially simple syntax for Java? If the statistical regularity of the local context being captured by the language model were arising solely from the simplicity of Java, then we should find this uniformly across all the projects; in particular, if we train a model on one Java project, and test on another, we should successfully capture the local regularity in the language. Thus, we reformulate the above question into the following:

RQ 2: *Is the local regularity that the statistical language model captures merely language-specific or is it also project-specific?*

For each of the 10 projects, we train a trigram model, and

evaluate its cross-entropy with each of the 9 others, and compare the value with the average 10-fold cross-entropy against itself. This plot is shown in Figure 2. The x-axis lists all the different Java projects, and, for each, the boxplot shows the range of cross-entropies with the other nine projects. The red line at the bottom shows the average self cross-entropy of the project against itself. As can be seen the self-entropy is always lower.

The language models capture a significant level of local regularity that is *not* an artifact of the programming language syntax, but rather arising from “naturalness,” or regularity, specific to each project.

This is a noteworthy result: it appears each project has its own type of local, non-Java-specific regularity that is being captured by the model; furthermore, the local regularity of each project is *special unto itself*, and different from that of the other projects. Most software engineers will find this intuitive: each project has its own vocabulary, and specific local patterns of iteration, field access, method calls, *etc.* It is important to note that the models are capturing non-Java-specific project regularity beyond simply the differences in unigram vocabularies. In Section 4, we discuss the application of the multi-token local regularity captured by the models to a completion task. As we demonstrate in that section, the models are able to successfully suggest non-linguistic tokens (tokens that are not Java keywords) about 50% of the time; this also provides evidence that the low entropy produced by the models are not just because of Java language simplicity.

But projects do not exist in isolation; the entire idea of *product-line engineering* rests on the fact that products in similar domains are quite similar to each other. This raises another interesting question:

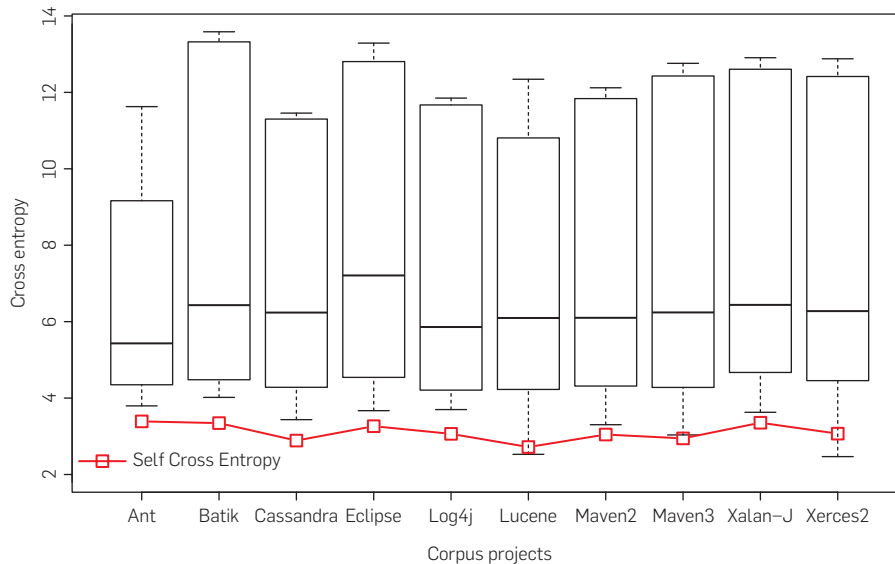
RQ 3: *Do n-gram models capture similarities within and*

differences between application domains?

We approached this question by studying application domains within Ubuntu. Table 1 lists the 10 the application domains we selected; these domains each contained a sizeable number of projects. The domains (with number of projects in each domain) are Administration (116), Documentation (22), Graphics (21), Interpreters (23), Mail (15), Networking (86), Sound/Audio (26), Tex/Latex related (135), Text processing (118), and Web (31). For each domain, we calculated the *self* cross-entropy within the domains, and the *other* cross-entropy, the cross-entropy of the domain against all the others. Cross-domain entropy (median ranging from 5 to 5.5 bits) is consistently and strongly higher, about a bit in most cases, than within-domain entropy (3–4.5 bits). Here again, as in the within-project versus between-project case, we see that there appears to be a lot of local regularity repeated *within* application domains, and much less so across application domains. Some domains, for example, the Web, appear to have much greater with-domain regularity (over 2 bits lower cross-entropy); this phenomenon merits further study.

Discussion We have seen that a high degree of local regularity is present in code corpora and, furthermore, that n -gram models effectively capture these local regularities. We find evidence suggesting that these regularities are specific to both projects and to application domains. We also find evidence that these regularities arise not merely from the relatively more regular (when compared to natural languages) syntax of Java, but also arise from other types of project- and domain-specific local regularities that exist in the code. In the next section, we demonstrate that these project-specific regularities are useful, by using project-specific models to extend the Eclipse suggestion engine; we also show that the n -gram models often (about 50% of the time) provide suggestions that are project-specific, rather than merely suggesting Java keywords guessable from context.

Figure 2. Ten projects cross-entropy versus self cross-entropy.



In natural language, these local regularities have proven to be of profound value, for tasks such as translation. It is our belief that these regularities can be used for code summarization and code searching; we also believe that deeper, semantic properties will also often manifest themselves in local regularities. These are discussed further in future work section (Section 6).

4. SUGGESTING THE NEXT TOKEN

The strikingly low entropy (between 3 and 4 bits) produced by the smoothed n -gram model indicates that, even at the local token-sequence level, there is a high degree of “naturalness.” If we make 8–16 guesses (2^3 – 2^4) as to what the next token is, we may very well guess the right one!

Eclipse Suggestion Plug-in We built an Eclipse plug-in to test this idea. Eclipse, like many other IDEs, has a built-in suggestion engine that suggests a next token whenever it can. Eclipse (and other IDEs) suggestions are typically based on type information available in context. We conjectured that corpus-based n -gram models suggestion engine (for brevity, $\mathcal{NGSE}$) could enhance eclipse’s built-in suggestion engine (for brevity, $\mathcal{ECSE}$) by offering tokens that tend to *naturally* follow from preceding ones in the relevant corpus.

The $\mathcal{NGSE}$ uses a trigram model built from the corpus of a lexed project. At every point in the code, $\mathcal{NGSE}$ uses the previous two tokens, already entered into the text buffer, and attempts to guess the next token. The language model built from the corpus gives maximum likelihood estimates of the probability of a specific choice of next token; this probability can be used to rank order the likely next tokens. Our implementation produces rank-ordered suggestions in less than 0.2s on average. Both $\mathcal{NGSE}$ and $\mathcal{ECSE}$ produce many suggestions. Presenting all would be overwhelming. Therefore we defined a heuristic to merge the lists from the two groups: given an admissible number n of suggestions

Algorithm 1. $\mathcal{MSE}$ (eprops, nprops, maxrank, minlen)

Require: eprops and nprops are ordered sets of Eclipse and N -gram proposals.

```

elong := {  $p \in \text{eprops}[1..\text{maxrank}] \mid \text{strlen}(p) > 6$  }
if elong  $\neq \emptyset$  then
    return eprops[1..maxrank]
end if
return eprops[1.. $\lceil \frac{\text{maxrank}}{2} \rceil$ ]  $\circ$  nprops[1.. $\lfloor \frac{\text{maxrank}}{2} \rfloor$ ]

```

to be presented to the user, choose n candidates from both $\mathcal{NGSE}$ ’s and $\mathcal{ECSE}$ ’s offers.

We observed that in general, $\mathcal{NGSE}$ was good at recommending *shorter* tokens, and $\mathcal{ECSE}$ was better at longer tokens (we discuss the reasons for this phenomenon later in this section). This suggested the simple merge algorithm $\mathcal{MSE}$, defined in Algorithm 1. Whenever Eclipse offers long tokens (we picked 7 as the break-even length, based on observation) within the top n , we greedily pick all the top n offers from Eclipse. Otherwise, we pick half from Eclipse and half from our n -gram model.

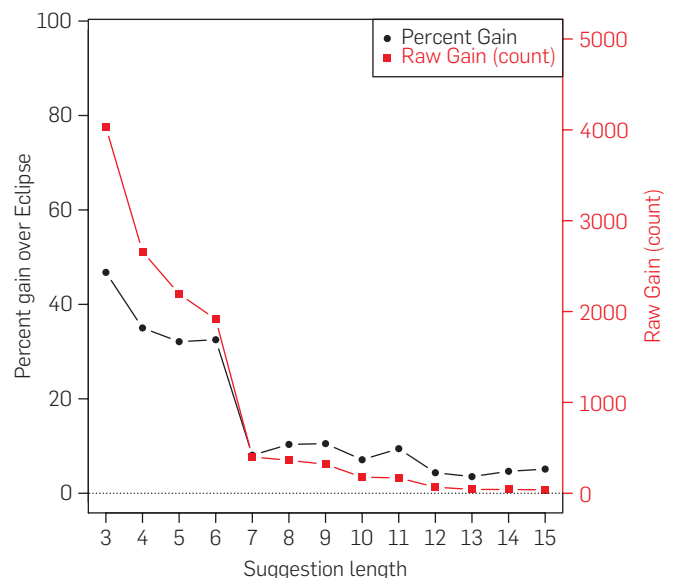
The relative performance of $\mathcal{MSE}$ and $\mathcal{ECSE}$ in actual practice might depend on a great many factors, and would

require a well-controlled, human study to be done at scale. A suggestion engine can present more or fewer choices; it may offer all suggestions, or only offer suggestions that are long. Suggestions could be selected with a touch-screen, with a mouse, or with a down-arrow key. Since our goal here is to gauge the power of corpus-based language models, as opposed to building the most user-friendly merged suggestion engine (which remains future work) we conducted an automated experiment, rather than a human-subject study.

We controlled for 2 factors in our experiments: the string length of suggestions l , and the number of choices n presented to the user. We repeated the experiment varying n , for $n = 2, 6, 10$ and l , for $l = 3, 4, 5, \dots, 15$. We omitted suggestions less than 3 characters, as not useful. Also, when merging two suggestion lists, we chose to pick at least one from each, and thus $n \geq 2$. We felt that more than 10 choices would be overwhelming—although our findings do not change very much at all even with 16 and 20 choices. We choose 5 projects for study: Ant, Maven, Log4J, Xalan, and Xerces. In each project, we set aside a test set of 40 randomly chosen files (200 set aside in all) and built a trigram language model on the remaining files for each project. We then used the $\mathcal{MSE}$ and $\mathcal{ECSE}$ algorithms to predict every token in the 40 set-aside files, and evaluated how many *more* times the $\mathcal{MSE}$ made a successful suggestion, when compared to the basic $\mathcal{ECSE}$. In each case, we evaluated the advantage of $\mathcal{MSE}$ over $\mathcal{ECSE}$, measured as the (percent and absolute) gain in number of correct suggestions at each combination of factors n and l .

How does the language model help? Figure 3 presents the results for the 6-suggestion case. The figure has two y-scales: the left side (black circle points) is percent additional correct suggestions, and the right side (red square points) are the raw counts. Since the raw count of successful suggestions from the Eclipse engine $\mathcal{ECSE}$ also declines with length, both measures are useful. $\mathcal{MSE}$ provides measurable advantage

Figure 3. Suggestion gains from merging n -gram suggestions into those of Eclipse.



over *ECSE* for 2, 6, as well as 10 suggestions, and for all token string lengths; however, the advantage in general declines with token length. The gains up through 6-character tokens are quite substantial, in the range of 33–67% additional suggestions from the language model that are correct; between 7 and 15 characters, the gains range from 3% to 16%. When compared to the 6 suggestions case (Figure 3), the gains are stronger with 2 suggestions and lower for 10 suggestions.

The additional suggestions from *NGSE* run the gamut, including methods, classes, and fields predictable from frequent trigrams in the corpus (e.g., `println`, `iterator`, `IOException`, `append`, `toString`, `assertEquals`, `transform`), package names (e.g., `apache`, `tools`, `util`, `java`) as well as language keywords (e.g., `import`, `public`, `return`, `this`). An examination of the tokens reveals why the *n*-grams approach adds most value with shorter tokens. The language model we build is based on *all* the files in the system, and the most frequent *n*-grams are those that occur frequently in all the files. In the corpus, we find that coders tend to choose shorter names for entities that are used more widely and more often; naturally these give rise to stronger signals that are picked up by the *n*-gram language model. It is worth repeating here that a significant portion, viz., 50% of the successful suggestions are not Java keywords guessed from language context—they are project-specific tokens. This reinforces our claim that the statistical language model is capturing a significant level of local regularity in the project corpus.

In the table below, we present a different way of looking at the benefit of *MSE*; the total number of keystrokes saved by using the base *ECSE*, (first row) the *MSE* (second row) and the percent gain from using *MSE*. Here, we used one specific language model to enhance one specific software tool, a suggestion engine. With more sophisticated techniques, that model syntactic, scoping, type, and semantic phenomena, we expect to achieve even lower entropy, and thus better performance in this and other software tools. We do invite readers to download our Eclipse Suggestion Plugin, *CACHECA*^f,¹¹ which based on a cache-enhanced locality-sensitive *n*-gram model,⁴⁹ and improved merging heuristics.

	Top 2	Top 6	Top 10
<i>ECSE</i>	42743	77245	95318
<i>MSE</i>	68798	103100	120750
Increase	61%	33%	27%

5. RELATED WORK

There are a few related areas of research, into which this line of work could be reasonably contextualized.

Code Completion and Suggestion *Completion* is the task of completing a partially typed-in token; *suggestion* suggests a complete token. Section 4 concerns *suggestion* engines.

Modern, mature software development environments (SDEs) provide both code completion and code suggestion, often with a unified interface. Two notable open-source Java-based examples are Eclipse and IntelliJ IDEA.

^f <http://macbeth.cs.ucdavis.edu/CACHECA>.

As in our work, these tools draw possible completions from existing code, but the methods of suggestion are fundamentally different.

Eclipse and IntelliJ IDEA respond to a programmer's completion request (a keyboard shortcut such as `ctrl+space`) by deducing what tokens “might apply” in the current syntactic context. Here, the tools are primarily guided by Java programming language semantics. For example, both Eclipse and IntelliJ IDEA respond to a completion request by parsing available source code and creating a short list of expected token types. If this list contains, say, a reference type, the tools use the rules of the type system to add a list of currently defined type-names to the list of completions. Similarly, if a variable is expected, the tools use names visible in the symbol table. Eclipse and IDEA implement dozens of these “syntactic and semantic rules” for various classes of syntactic constructs. As a final step, both tools rank the completions with a collection of apparently hand-coded heuristics.

Our approach is complementary. Rather than using language semantics and immediate context, to guess what *might* apply, our *n*-gram language model captures what *most often does* apply. Our approach offers a great deal more flexibility and it also has the potential to be much more precise: the space of commonly used completions is naturally far smaller than the space of “language-allowed” completions. Further, our approach can enhance (or order) semantically informed suggestions and completions. It is noteworthy that perhaps some of the strongest evidence for the “naturalness” of software, though, is how well our completion prototype functions *in spite of not using semantic information*.

There are approaches arguably more advanced than those currently available in IDEs. The BMN completion algorithm of Bruch et al.⁸ focuses on finding the most likely *method calls* likely to complete an expression by using frequency of method calls and prior usage in similar circumstances. We propose a broad vision for using language models of code corpora in software tools. Our specific illustrative completion application has a broader completion goal, completing all types of tokens, not just method calls. Later, Bruch et al.⁷ lay out a vision for next-generation IDEs that take advantage of “collective wisdom” embodied in corpora and recorded human action. We enthusiastically concur, and suggest the use of language models from NLP to capture this wisdom.

Robbes and Lanza⁴² compare a set of different method-call and class-name completion strategies, which start with a multi-letter prefix. They introduce an approach based on history and show that it improves performance. Our approach can complement history-based approaches by adding a language model. Han et al.¹⁴ uses Hidden Markov Models (HMM) to infer likely tokens from abbreviations. They make use of a dynamic programming trellis approach for back-tracking and suggestion. Their HMM is, in fact, a language model, but the paper does not describe how effective it is or how well it would perform for completion tasks without user-provided abbreviations.

Jacob and Tairas¹⁹ used *n*-gram language models for a different application: to find matching code clones relevant to partial programming task. Language models were built over

clone groups (not entire corpora as we propose) and used to retrieve and present candidate clones relevant to a partially completed coding task.

Hou and Pletcher¹⁷ used context-specific API information in order to sort Eclipse code completions using static types. Users can filter out future completion proposals.

The “Naturalness” of Names in Code A line of work has been automatically evaluating the quality of entity names in code, asking “Do the names reflect the meanings of the artifacts, and if not, how could they be improved^{6, 27?}” Work by Høst and Østfold^{15, 16} also concerns method naming; they combine static analysis with an entropy-based measure over the distribution of simple semantic properties of methods in a corpus to determine which method names are most discriminatory, and they use it to detect names whose usage is inconsistent with the corpus.

Summarization and Concern Location A line of work on code summarization^{9, 47, 48} uses semantic properties inferred by static analysis, rather than the “natural” regularities of software corpora capturable in statistical models; it is complementary to ours: properties derived by static analysis (as long as they can be done efficiently, and at scale) could be used to enrich statistical models of large software corpora. Another line of work seeks to locate parts of code relevant to a specified concern (e.g., “place auction bid”), which could be local or cross-cutting, based on fragments of code names,⁴⁴ facts mined from code,⁴⁰ or co-occurrence of related words in code.⁴⁵

NLP and Requirements Researchers have studied the use of natural language in requirements engineering: “Can we utilize natural language specifications to automatically generate more formal specifications, or even code?”^{25, 26, 43} Some of these approaches make use of NLP tools like parsers and part-of-speech taggers. Our approach considers instead the “naturalness” of *code*, as embodied in statistical models built from large corpora of artifacts; however we also consider (in future work) the possibility of using aligned (code/natural language) corpora for tasks such as code summarization or code search, which could be viewed as translation tasks.

Software Mining Work in this very active area⁵¹ aims to mine *useful information* from software repositories. Many papers can be found in MSR conference series at ICSE, and representative works include mining API usages,^{12, 30} patterns of errors,^{21, 29} topic extraction,²⁸ guiding changes,⁵² and several others. The approaches used vary. We argue that the “naturalness” of software provides a *conceptual perspective* for this work, and also offers some novel *implementation* approaches. The *conceptual perspective* rests on the idea that *useful information* is often manifest in software in uniform, and uncomplicated ways; the *implementation approach* indicates that the uniform and uncomplicated manifestation of useful facts can be determined from a large, representative software corpus in which the required information is already known and annotated. This corpus can be used to estimate the statistical relationship between the required information and readily observable facts; this relationship can be used to reliably find similar information in new programs similar to the corpus. We explain this further in future work (Sections 6.3 and 6.4).

6. FUTURE DIRECTIONS

Statistical models estimated over large code corpora hold tremendous promise in software engineering applications. We list some applications below.

6.1. Improved language models

In this paper, we exploited a common language model (*n*-grams) that effectively captures local regularity. There are several avenues for extension. Existing, very large bodies of code can be readily parsed, typed, scoped, and even subject to simple semantic analysis. All of these can be modeled using enhanced language models to capture regularities that exist at syntactic, type, scope, and even semantic levels.

There is a difficulty here: the richer a model, the more data is needed to provide good estimates for the model parameters; thus, the risk of data sparsity grows as we enrich our models. Ideas analogous to the smoothing techniques used in *n*-gram models will have to be adapted and applied to build richer models of software corpora. In follow-on work, researchers have exploited type models,³⁸ syntactic models,^{1, 35} scope,⁴⁹ and simple data flow information,⁴¹ for various tasks. Recent work has also explored deep learning approaches to modeling code.⁵⁰ Regularities in syntax, type, scope, and even semantic properties can be captured in models and exploited. For example, Campbell et al.¹⁰ diagnose syntax error locations by exploiting the fact that language models trained on working code will be surprised by syntax errors. Indeed there are a great many potential applications to software engineering tasks, some of which we discuss further below.

6.2. Language models for accessibility

Some programmers have difficulty using keyboards, because of RSI or visual impairment. Prior work explored using speech recognition (e.g., Refs.^{3, 4, 18}) for assistance. However, due to fairly high recognition error rates, adoption has been limited.³³ None of the published approaches make use of a statistical language model trained on specific code corpora. The use of a code-specific language model will surely reduce error rates, as in conventional speech recognition engines. Much development work occurs in a maintenance or re-engineering context, where ample data is available for model estimation. Even when only a small amount of relevant code exists, language model adaptation,⁵ and cacheing⁴⁹ could be applied.

6.3. Applications of machine translation

Consider the task of summarizing a fragment of code (or change to code) in English. Consider also the approximate reverse task: finding/retrieving a relevant fragment of code (e.g., method call) given an English description. We draw an analogy between these two problems and *statistical natural language translation (SNLT)*. Code and English are two languages, and essentially both the above are *translation* tasks. *SNLT* relies on access to an *aligned corpus*, which is a large set of sentences simultaneously presented in two or more languages (e.g., parliamentary proceedings, or Scriptural verses). Consider the problem of translating a Tamil sentence *T* to an English sentence *E*. *SNLT* uses a simple Bayesian formulation to calculate the likeliest English translation; it

maximizes the right hand side of this equation over all possible English sentences E :

$$p(E|T) = \frac{p(T|E)p(E)}{p(T)}$$

For a given T , the denominator is constant, and we just maximize the numerator. Within the numerator, the conditional distribution $p(T|E)$ is estimated using E - T pairs in an aligned corpus; $p(E)$ is provided by a language model estimated over an English corpus. The formulation is clearly also applicable for the reverse translation task. The analogy to code summarization/retrieval tasks is evident: given a code fragment C , produce an English summary E ; or, conversely, retrieve a code fragment C given an English query E . For the former (summarization) task, we need the conditional distribution $p(C|E)$, and a language model $p(E)$.

First, how do we estimate the conditional distribution $p(C|E)$? Several promising sources of aligned (English/Code) corpora exist. Consider the version history of a project. Each commit potentially offers a matched pair of a log message (English), and some changes (Code). Second, many in-line comments could be matched with nearby code.² Stack-overflow posts (both questions and answers) often include closely-related code and text. Next, consider $p(E)$: the relevant (natural language) corpora concerning the code can be found in project-associated English language text, for example, code comments, documents, bug reports, mailing-list discussions, etc. We can thus estimate $p(E)$. These $p(C|E)$ and $p(E)$ models can then select maximally likely code fragments given English descriptions as shown above; the reverse task would use $p(E|C)$ and $p(C)$ models, which are also analogously estimable. This approach could use semantic properties of code, as well; however, unlike Buse and Weimer,⁹ who document the semantics of each change in isolation, we would use statistics of semantic properties over large aligned code/text corpora.

Porting between languages or dialects is another potential application; Nguyen and colleagues^{34, 36, 37} have exploited the availability of cross-platform applications in Java and C#, with method-level alignments, to train translation models, and learn API mappings. Karaivanov et al.²⁰ have followed the above work in a similar vein.

6.4. Software tools

We hypothesize that the “naturalness” of software implies a “naturalness” of deeper properties of software, such as those normally computed by powerful but expensive software tools; we hypothesize that (because programs tend to be repetitive) deeper, more semantic properties of programs are *also* manifest in programs in superficially similar ways. More specifically, we hypothesize that semantic properties are *usually* manifest in *superficial* ways that are computationally cheap to detect, particularly when compared to the cost (or even infeasibility) of determining these properties by sound (or complete) static analysis. If feasible, one can leverage this notion to build simple, scalable, and effective approximations in a wide variety of settings, as we now describe.

Consider a very general formulation of a software tool

that computes a function f over a system s , drawn from a domain of systems $\mathcal{S}$ thus:

$$f : \mathcal{S} \rightarrow \mathcal{D}$$

where the range $\mathcal{D}$ represents a fact derived by analysis, for example, a may-alias fact, a mined rule (e.g., an API protocol), a error message (e.g., buffer-overflow warning), or even a new, transformed version of the input system $s \in \mathcal{S}$. Let us assume that the function f is, in general, expensive or even infeasible to calculate, and is thus subject to approximation error and calculation delays in practice. We reformulate this problem by estimating f using an alternative formulation, $\hat{f}$, which chooses the *most likely* value of $f(s)$, given an easily calculated and readily observed set of *evidence features* $e_1(s)$, $e_2(s)$, ... Features $e_i(s)$ are “symptoms”; they are not deterministic, conclusive proof that $f(s)$ has some property ϕ , but nevertheless provide varying levels of justification for a probabilistic belief in ϕ .

This view has a Bayesian formulation. Denote the probability that d holds of system s , given the observed bits of evidence $e_i(s)$:

$$p(d|s) = \frac{p(e_1(s), e_2(s), \dots, e_n(s)|d)p(d)}{p(e_1(s), e_2(s), \dots, e_n(s))}$$

The distributions on the right-hand side can be estimated over large corpora. Given a specific system, the denominator is a constant. For the numerator, the prior distribution $p(d)$, over the output domain of f , can be estimated from observations in the corpus (e.g., “How often do buffer overflow errors occur?” “How often do different protocol patterns occur?” etc.). In the absence of such information, a uninformative (uniform) prior could be chosen.

Next, we estimate the strength of the association (likelihood) between the conclusion d and the observed features $e_1(s) \dots e_n(s)$ in the corpus. $p(e_1(s), e_2(s), \dots, e_n(s)|d)$ can be viewed as the conditional frequency of the observation of features in the corpus when the output domain property d holds. This term requires the availability of a corpus annotated with property d . Even in cases where d must be manually annotated, we argue that such corpora may be well-worth the investment (by analogy with the Penn Tree Bank³²) and can be constructed using volunteer open-source community, and perhaps market mechanisms like the Mechanical Turk.²²

Finally, assuming suitable models to enable calculation of $p(d|s)$ can be estimated, we write the alternative, probabilistic function $\hat{f}$ to choose the most likely d :

$$\hat{f}(s) = \arg \max_{d \in \mathcal{D}} p(d|s)$$

We hasten to add that this Bayesian formulation is just to provide an intuition on how corpus-based statistics can be used in a principled way to aid in the construction of approximating software engineering tools. In practice, a machine learning formulation (such as decision-trees, support-vector machines, or even simple regression models) might prove more expedient.

Raychev et al.⁴¹ demonstrated this idea, with an application to guessing useful names of variables in obfuscated

code. Variable names are chosen by programmers to represent semantic properties; a corpus of code with clear names can be used for training. Using very simple data-flow properties (readily derivable from syntax) as predictive features, and a log-linear formulation of the above idea, they were able to guess helpful variable names.

7. CONCLUSION

Although Linguists (sometimes) revel in the theoretical complexities of natural languages, most “natural” utterances, in practice, are quite regular and predictable and can in fact be modeled by rigorous statistical methods. This fact has revolutionized computational linguistics. We offer evidence supporting an analogous claim for software: *though software in theory can be very complex, in practice, it appears that even a fairly simple statistical model can capture a surprising amount of regularity in “natural” software*. This simple model is strong enough for us to quickly and easily implement a fairly powerful suggestion engine that already improves a state-of-the-art IDE. We also lay out a vision for future work. Specifically, we believe that natural language translation approaches can be used for code summarization and code search in a symmetric way; we also hypothesize that the “naturalness” of software implies a sort of “naturalness” of deeper properties of software, such as those normally computed by powerful, traditional software analysis tools. These are challenging tasks, but with potentially high pay-off, and we hope others will join us in this work.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 1445079, 1247280, 1414172.

References

- Allamanis, M., Sutton, C. Mining idioms from source code. In *FSE*. ACM, 2014.
- Antoniol, G., Canfora, G., Casazza, G., Lucia, A.D., Merlo, E. Recovering traceability links between code and documentation. *IEEE Trans. Softw. Eng.* 28 (2002), 970–983.
- Arnold, S., Mark, L., Goldthwaite, J. Programming by voice, VocalProgramming. In *Proceedings, ACM Conference on Assistive Technologies*. ACM, 2000, 149–155.
- Begel, A. Spoken language support for software development. In *Proceedings, VL/HCC*. IEEE Computer Society, 2004, 271–272.
- Bellegarda, J. Statistical language model adaptation: Review and perspectives. *Speech Commun.* 42, 1 (2004), 93–108.
- Binkley, D., Hearn, M., Lawrie, D. Improving identifier informativeness using part of speech information. In *Proceedings, MSR*. ACM, 2011.
- Bruch, M., Boddien, E., Monperrus, M., Mezini, M. Ide 2.0: Collective intelligence in software development. In *Proceedings of the FSE/SDP workshop on Future of Software Engineering Research*. ACM, 2010, 53–58.
- Bruch, M., Monperrus, M., Mezini, M. Learning from examples to improve code completion systems. In *Proceedings, ACM SIGSOFT ESEC/FSE*, 2009.
- Buse, R., Weimer, W. Automatically documenting program changes. In *Proceedings, ASE*. ACM, 2010, 33–42.
- Campbell, J.C., Hindle, A., Amaral, J.N. Syntax errors just aren't natural: Improving error reporting with language models. In *MSR*. ACM, 2014.
- Franks, C., Tu, Z., Devanbu, P., Hellendoorn, V. Cacheca: A cache language model based code suggestion tool. *ICSE Demonst. Track* (2015).
- Gabel, M., Su, Z. Javert: Fully automatic mining of general temporal properties from dynamic traces. In *Proceedings, ACM SIGSOFT FSE*. ACM, 2008, 339–349.
- Gabel, M., Su, Z. A study of the uniqueness of source code. In *Proceedings, ACM SIGSOFT FSE*. ACM, 2010, 147–156.
- Han, S., Wallace, D.R., Miller, R.C. Code completion from abbreviated input. In *ASE*. IEEE Computer Society, 2009, 332–343.
- Høst, E.W., Østfold, B.M. *Software Language Engineering. Chapter The Java Programmer's Phrase Book*. Springer-Verlag, Berlin, Heidelberg, 2009.
- Høst, E., Østfold, B. Debugging method names. In *Proceedings ECOOP*. Springer, 2009, 294–317.
- Hou, D., Pletcher, D. An evaluation of the strategies of sorting, filtering, and grouping API methods for code completion. In *Proceedings, ICSM*, 2011.
- Hubbell, T., Langan, D., Hain, T. A voice-activated syntax-directed editor for manually disabled programme RS. In *Proceedings, ACM SIGACCESS*. ACM, 2006.
- Jacob, F., Tairas, R. Code template inference using language models. In *Proceedings of the 48th Annual Southeast Regional Conference*, 2010.
- Karavayev, S., Raychev, V., Vechev, M. Phrase-based statistical translation of programming languages. In *Onward!*. ACM, 2014, 173–184.
- Kim, S., Pan, K., Whitehead, E. Jr. Memories of bug fixes. In *Proceedings, ACM SIGSOFT FSE*. ACM, 2006, 35–45.
- Kittur, A., Chi, E., Suh, B. Crowdsourcing user studies with mechanical turk. In *Proceedings, CHI*. ACM, 2008.
- Knuth, D.E. Literate programming. *Comput. J.* 21, 2 (1984), 97–111.
- Koehn, P. *Statistical Machine Translation*. Cambridge University Press, 2010.
- Konrad, S., Cheng, B. Real-time specification patterns. In *Proceedings ICSE*, 2005.
- Körner, S.J., Tichy, W.F. Text to software. In *Proceedings of FSE/SDP Workshop on the Future of Software Engineering Research*. ACM, Nov. 2010.
- Lawrie, D., Morrell, C., field, H., Binkley, D. What's in a name? A study of identifiers. *Proceedings, ICPC*, 2006.
- Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C., Baldi, P. Sourcecr: Mining and searching internet-scale software repositories. *Data Mining Knowl. Discov.* 18, 2 (2009), 300–336.
- Livshits, B., Zimmermann, T. Dynamine: Finding common error patterns by mining software revision histories. *ACM SIGSOFT Softw. Eng. Notes* 30, 5 (2005).
- Mandelin, D., Xu, L., Bodik, R., Kimelman, D. Jungloid mining: Helping to navigate the API jungle. In *ACM SIGPLAN Notices*. Volume 40. ACM, 2005.
- Manning, C., Schütze, H. *Foundations of Statistical Natural Language Processing*. Volume 59. MIT Press, 1999.
- Marcus, M., Marcinkiewicz, M., Santorini, B. Building a large annotated corpus of English: The Penn treebank. *Comput. Linguist.* 19, 2 (1993), 313–330.
- Mills, S., Saadat, S., Whiting, D. Is voice recognition the solution to keyboard-based RSI? In *Automation Congress, 2006. WAC'06. World*, 2006.
- Nguyen, A.T., Nguyen, H.A., Nguyen, T.T., Nguyen, T.N. Statistical learning approach for mining API usage mappings for code migration. In *ASE*, 2014.
- Nguyen, A.T., Nguyen, T.N. Graph-based statistical language model for code. 2015.
- Nguyen, A.T., Nguyen, T.T., Nguyen, T.N. Lexical statistical machine translation for language migration. In *FSE*, 2013.
- Nguyen, A.T., Nguyen, T.T., Nguyen, T.N. Migrating code with statistical machine translation. In *ICSE Companion*, 2014.
- Nguyen, T.T., Nguyen, A.T., Nguyen, H.A., Nguyen, T.N. A statistical semantic language model for source code. In *ESEC FSE*. ACM, 2013.
- Pinker, S. *The Language Instinct: The New Science of Language and Mind*. Volume 7529. Penguin, London, UK, 1994.
- Rastkar, S., Murphy, G.C., Bradley, A. Generating natural language summaries for cross-cutting source code concerns. In *Proceedings, ICSM*, 2011.
- Raychev, V., Vechev, M., Krause, A. Predicting program properties from big code. In *POPL*. ACM, 2015.
- Robbes, R., Lanza, M. Improving code completion with program history. *Autom. Softw. Eng.* 17, 2 (2010), 181–212.
- Rolland, C., Proix, C. A natural language approach for requirements engineering. In *Advanced Information Systems Engineering*. Springer, 1992, 257–277.
- Shepherd, D., Fry, Z., Hill, E., Pollock, L., Vijay-Shanker, K. Using natural language program analysis to locate and understand action-oriented concerns. In *Proceedings, AOSD*. ACM, 2007, 212–224.
- Shepherd, D., Pollock, L., Tourwé, T. Using language clues to discover crosscutting concerns. In *ACM SIGSOFT Software Engineering Notes*. Volume 30. ACM, 2005.
- Sparck-Jones, K. Natural language processing: A historical review. *Current Issues in Computational Linguistics: In Honour of Don Walker* (Ed Zampolli, Calzolari and Palmer). Kluwer, Amsterdam, the Netherlands, 1994.
- Sridhara, G., Hill, E., Muppaneni, D., Pollock, L., Vijay-Shanker, K. Towards automatically generating summary comments for Java methods. In *Proceedings, ASE*, 2010.
- Sridhara, G., Pollock, L., Vijay-Shanker, K. Automatically detecting and describing high level actions within methods. In *Proceedings, ICSE*, 2011.
- Tu, Z., Su, Z., Devanbu, P. On the localness of software. In *FSE*. ACM, 2014, 269–280.
- White, M., Vendome, C., Linares-Vásquez, M., Poshvanyk, D. Toward deep learning software repositories. In *MSR*, 2015.
- Xie, T., Thummalapenta, S., Lo, D., Liu, C. Data mining for software engineering. *IEEE Comput.* 42, 8 (2009).
- Zimmermann, T., Weisgerber, P., Diehl, S., Zeller, A. Mining version histories to guide software changes. In *ICSE*. IEEE Computer Society, 2004.

Abram Hindle (abram.hindle@ualberta.ca), Department of Computing Science, University of Alberta, Edmonton, Canada.

Earl T. Barr (e.barr@ucl.ac.uk), Department of Computer Science, University College London, United Kingdom.

Mark Gabel, Zhendong Su, and Premkumar Devanbu (mark.gabel@gmail.com, [su, ptdevanbu]@cs.ucdavis.edu), Department of Computer Science, UC Davis, CA.

CAREERS

Dartmouth College Computer Science Dept Lecturer or Senior Lecturer

The Department of Computer Science at Dartmouth College invites applications for a Lecturer or Senior Lecturer position. We seek individuals interested in teaching systems-oriented courses including—but not necessarily limited to—Object Oriented Programming, Software Engineering, Smartphone Programming, and Web Programming. Lecturers at Dartmouth are valued members of faculty and are provided with the resources needed to teach at the high standard expected of the faculty. Dartmouth operates on a quarter-system and the teaching load for this position is 4 courses per year.

Dartmouth's Computer Science department is home to 22 faculty with strong and growing Ph.D., M.S. and undergraduate programs. Dartmouth College is located in Hanover, New Hampshire (on the Vermont border). Dartmouth has a beautiful, historic campus, located in a scenic area on the Connecticut River. Recreational opportunities abound in all four seasons.

The position requires a Ph.D. in Computer Science or related discipline (applicants with an

MS degree and significant teaching experience will also be considered). The appointment begins September 1, 2016. This position is a 1 year appointment, with a possibility of extending to 1-2 more years. The level of appointment and compensation will be commensurate with the applicant's qualifications.

APPLY FOR THIS JOB: Contact: Joseph Elsener, Email: joseph.elsener@dartmouth.edu, Phone: 603-646-1358

Princeton University Computer Science Full-Time Lecturer(s)

The Department of Computer Science seeks applications from outstanding individuals who share our strong commitment to undergraduate education to join our teaching faculty.

Computer Science is enjoying record popularity at Princeton, and opportunities abound to engage with our outstanding students at many levels. Our large undergraduate courses are the shared responsibility of a team of faculty and graduate assistants. A successful candidate for

this position will participate in such a team at the outset. After gaining experience in this way, job responsibilities can include teaching upper-level courses, advising undergraduate research, curriculum development, state-of-the-art software technology development, data analytics, outreach to under-represented groups, and online content development.

Research and scholarship in CS education or in any area of CS is also encouraged. An advanced degree in computer science, or a related field, is required (PhD preferred).

Further information about the Computer Science Department at Princeton can be found at: <http://cs.princeton.edu/>.

Applications must include a cover letter, curriculum vitae, teaching statement, material relevant to evaluating the applicant's teaching abilities and research accomplishments, and the names of at least three references. To apply, please submit a cover letter, CV, and contact information for three references to (Princeton University). This position is subject to the University's background check policy.

Princeton University is an Equal Opportunity/Affirmative Action Employer and all qualified ap-



TENURE-TRACK AND TENURED POSITIONS IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

The newly launched ShanghaiTech University is built as a world-class research university, which locates in Zhangjiang High-Tech Park. We invite highly qualified candidates to fill tenure-track/tenured faculty positions as its core team in the School of Information Science and Technology (SIST). Candidates should have exceptional academic records or demonstrate strong potential in cutting-edge research areas of information science and technology. They must be fluent in English. Overseas academic connection or background is highly desired.

ShanghaiTech is matching towards a world-class research university as a hub for training future generations of scientists, entrepreneurs, and technological leaders. Located in a brand new campus in Zhangjiang High-Tech Park of the cosmopolitan Shanghai, ShanghaiTech is at the forefront of modern education reform in China.

Academic Disciplines: We seek candidates in all cutting edge areas of information science and technology. Our recruitment focus includes, but is not limited to: computer architecture and technologies, nano-scale electronics, high speed and RF circuits, intelligent and integrated signal processing systems, computational foundations, big data, data mining, visualization, computer vision, bio-computing, smart energy/power devices and systems, next-generation networking, as well as inter-disciplinary areas involving information science and technology.

Compensation and Benefits: Salary and startup funds are highly competitive, commensurate with experience and academic accomplishment. We also offer a comprehensive benefit package to employees and eligible dependents, including housing benefits. All regular ShanghaiTech faculty members will be within its new tenure-track system commensurate with international practice for performance evaluation and promotion.

Qualifications:

- A detailed research plan and demonstrated record/potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, or related field);
- A minimum relevant research experience of 4 years.

Applications: Submit (in English, PDF version) a cover letter, a 2-page research plan, a CV plus copies of 3 most significant publications, and names of three referees to: sist@shanghaitech.edu.cn (until positions are filled). For more information, please visit Job Opportunities on <http://sist.shanghaitech.edu.cn/>

Deadline: April 30th, 2016



KUWAIT COLLEGE OF SCIENCE AND TECHNOLOGY OPENINGS FOR TENURE TRACK FACULTY

Kuwait College of Science and Technology (KCST), is a new University with state-of-the-art infrastructure, focussing primarily on three programs leading to the Bachelor's Degree: Computer Science, Computer Engineering and Electronics and Communications Engineering. KCST invites applications from highly qualified and motivated candidates for tenure track positions of **Assistant Professor, Associate Professor and Professor** in the above programs, effective as of **August 1, 2016**. The successful candidates are expected to demonstrate a commitment to teaching at the undergraduate and graduate levels, student support, research and graduate student supervision. Furthermore, they are expected to engage in scholarly activities, service to the University and professional societies.

1. Department of Electronics and Communications Engineering:

Candidates must have earned a Ph.D. degree in above fields and must show relevant experience, corresponding to the level of the position, in teaching and research, all in reputed institutions. Preference will be given to candidates with strong record in the following areas: **Circuits and Systems, Microelectronics, Signal Processing, Electronic Devices, VLSI, Communications and Embedded Systems.**

2. Departments of Computer Science and Computer Engineering:

Candidates must have earned a Ph.D. degree in the above fields and must show relevant experience, corresponding to the level of the position, in teaching and research, all in reputed institutions. Preference will be given to candidates with strong record in the following areas: **Programming Languages/Compilers, Computer Organization and Architecture, Computer Networks, Software Engineering, Data Bases and Operating Systems.**

Information about the College can be found at <http://www.kcst.edu.kw/>. **KCST** offers competitive salary packages, various benefits and allowances.

All qualified candidates are encouraged to apply. Review of applications will begin immediately, and continue until the positions are filled. For full consideration, applicants should submit preferably a **single ZIP file** containing: (i) cover letter, (ii) curriculum vitae, (iii) statement of research (3 pages maximum) and a list of recent publications, (iv) philosophy of teaching (2 pages maximum) and (v) list of three professional references with contact information **to faculty.jobs@kcst.edu.kw**.

plicants 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.

You may apply online at: <https://jobs.cs.princeton.edu/lecturer/>

Requisition Number: 1600161

Qatar University

Assistant/Associate/Full Research Professor in Cyber Security

Qatar University invites applications for research faculty positions at all levels.

Candidates will cultivate and lead research projects at the KINDI Center for Computing Research in the area of Cyber Security.

Qatar University offers competitive benefits package including a 3-year renewable contract, tax free salary, free furnished accommodation, and more.

Apply by posting your application on the QU online recruitment system at careers.qu.edu.qa under "College of Engineering"

University of Central Missouri

Assistant Professor of Computer Science - Non Tenure Track

The Department of Computer Science at the University of Central Missouri (UCM) invites applications for several non-tenure track assistant

professor positions in Computer Science to begin in August 2016.

Applicants must have Ph.D. degree in Computer Science or a closely related area. ABD can be considered. Previous college/university teaching experience is highly desirable. The successful candidate should be able to teach a variety of undergraduate and graduate computer science courses.

The faculty will be expected to teach 12 credit hours per semester at UCM's Warrensburg and/or Lee's Summit campus.

To apply online, go to <https://jobs.ucmo.edu>. Apply to position #997495. The following items should be attached: a letter of interest, a curriculum vitae, copies of transcripts, and a list of at least three professional references including their names, addresses, telephone numbers and email addresses. Contact: Dr. Songlin Tian. Email: tian@ucmo.edu

Initial screening of applications begins March 15, 2016, and continues until position is filled. AA/EEO/ADA. Women and minorities are encouraged to apply.

University of Vienna TU Wien, and Austrian Institute of Science and Technology

PhD candidates in Vienna Graduate School in Computational Optimization

The Vienna Graduate School in Computational Optimization offers Prae-Doc contracts for PhD candidates in different areas of optimization such as Combinatorial Optimization, Global Op-

timization, Heuristic Optimization, Nonlinear Optimization, Stochastic Optimization, Dynamic Optimization, Algorithmic Game Theory. Application deadline: 05/31/2016. See <http://vgsco.univie.ac.at>. Email: vsgco@univie.ac.at

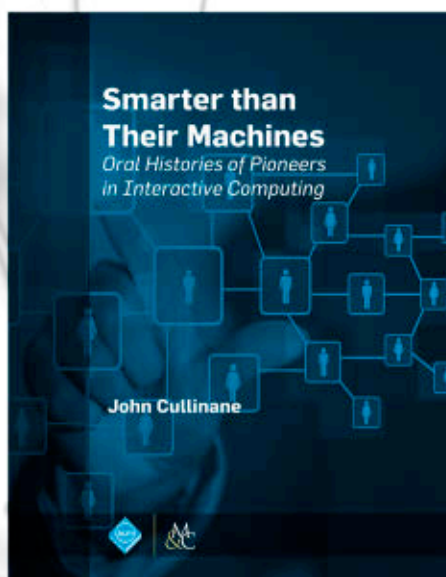
University of Wisconsin – Stevens Point Department of Computing and New Media Technologies Assistant/Associate Professor

The Department of Computing and New Media Technologies at UW-Stevens Point invites applications for the Sentry Insurance Endowed Chair in Computational Analytics, a tenure-track faculty position starting August 2016. We are particularly interested in candidates with skills in:

- ▶ data validation, cleansing and transformation
- ▶ data acquisition and storage
- ▶ statistical data analysis and data mining
- ▶ data visualization and dissemination

Required Ph.D. in data analytics, data science, or related field. The candidate is expected to teach undergraduate courses in a Bachelor's program in Data Analytics. There may be opportunity to teach graduate courses in an online Master's program in Data Science. See <http://www.uwsp.edu/hr/jobs/Pages/default.aspx> for application instructions and additional information on our department.

Screening of applicants will begin on April 15, 2016 and continue until the position is filled. UW-Stevens Point is an Affirmative Action/Equal Opportunity Employer.



A personal walk down the computer industry road. BY AN EYEWITNESS.

Smarter Than Their Machines: Oral Histories of the Pioneers of Interactive Computing is based on oral histories archived at the Charles Babbage Institute, University of Minnesota. These oral histories contain important messages for our leaders of today, at all levels, including that government, industry, and academia can accomplish great things when working together in an effective way.



ISBN: 978-1-62705-550-5 DOI: 110.1145/2663015

<http://books.acm.org>

<http://www.morganclaypoolpublishers.com/acm>

Are you looking for your next IT job?

Do you need Career Advice?

The **ACM Career & Job Center** offers ACM members a host of career-enhancing benefits:

- A **highly targeted focus** on job opportunities in the computing industry
- **Access to hundreds** of industry job postings
- Resume posting **keeping you connected** to the employment market while letting you maintain full control over your confidential information
- **Job Alert system** that notifies you of new opportunities matching your criteria
- **Career coaching** and guidance available from trained experts dedicated to your success
- **Free access** to a content library of the best career articles compiled from hundreds of sources, and much more!



Visit **ACM's Career & Job Center** at:
<http://jobs.acm.org>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

The **ACM Career & Job Center** is the perfect place to begin searching for your next employment opportunity!

Visit today at **<http://jobs.acm.org>**

[CONTINUED FROM P. 136] round-trip flight around Mars five years after that (2033), followed by the initiation of a Mars orbit outpost in 2038; and a landing on the surface six years later in 2044. The Mars goal laid out by President Barack Obama in 2010 following the report of his Review of U.S. Human Spaceflight Plans Committee in 2009^a had proved sustainable, as well as internationally appealing, with nearly 20 nations represented on the three bases now on the Red Planet. The bases were still outposts, dependent on Earth supplies, but in the 25 years since the first landing on Mars their use of indigenous Martian resources had increased to the point that power, propulsion, and water were completely Martian-supplied and 25% of the food was locally grown. Oxygen, along with manufactured supplies, was imported from Earth, though less and less so.

Dr. Okonjo would give her speech from inside a domed amphitheater at the Tharsis Bulge near Mars's equatorial region. Behind her was Olympus Mons, the tallest mountain in the solar system, the backdrop chosen for the amphitheater locale. Most human activity was conducted underground or in protective shelters, but having Martian vistas was important to the explorers whose goal was planetary settlement. She reflected, "In a time period shorter than between Columbus's first landing on a small Western Hemisphere island to the beginning of the first European settlement in North America 111 years later, we had moved from the first human landing on another world to the beginning of a new-world settlement. May our vision for Mars be even more bright and far-reaching than for America four centuries ago."

At the same time, 2069, back on Earth, another explorer, Carlos Gupta, is stepping through a portal in his home office directly into his AI display. There, information is processed from data streaming back from robotic European submarines. It will be organized into holographic virtual-world displays for human observation, experience, and interaction, convincingly real. The data

I see no contradiction or conflict in these two paths for human-robotic evolution—on Mars physically, in the Universe virtually.

comes from the six submarines now roaming the European ocean, plus three more on Europa's icy surface, serving as both backup vehicles to the explorers below and as surface observation and communication outposts. Included are optical images, visible, infrared, and ultraviolet; sound at a range of frequencies spanning several times the human range; mass spectrometer and chemical data providing the equivalent of olfactory and taste data; and even tactile data from the many appendages and external sensors on the submersibles.

The data processed into the classical five senses of information by the AI programs in Gupta's computers provides the five classical human senses of sight, hearing, smell, taste, and touch to the explorers in the virtual world created through the display. The AI also provides the arguably not-so-classical sixth sense of intuition to help humans in the virtual world infer an equivalent real-world experience, including observation, inference, hypothesis, conclusions, and predictions.

Gupta can reflect, with an irony opposite of Okonjo's on Mars, on being a stay-at-home explorer of other worlds. With his training as an experimental and field astrobiologist working in extreme environments, Gupta would have expected to have the experience of being physically off-Earth in a new world, not exploring virtually in his own home. But in less than a century after a large subsurface European water ocean was first detected, he was able to immerse himself in that ocean to search for, maybe even interact with, putative extraterrestrial life there.

The virtual world Gupta explores is also available to millions more on Earth, creating a sense of being there

that enables human presence far beyond the limits of physical human travel. Rather than wish for non-invented technologies or devices that break the laws of physics for human space travel to a theoretically habitable world around another star, say, 100 trillion kilometers from Earth, we get there much sooner in virtual worlds created through real data and real human interaction. A nano-spacecraft loaded with nano- or even pico-technology sensors flying at enormously high velocity propelled by lightsails pushed by large laser arrays in our solar system can stream data from an exoplanet to earthbound supercomputers, or AI machines. We will create a virtual world in which we extend our human presence. Even in this century we can do this at the outer planets of the solar system, without the kind of laser arrays needed for interstellar flight. Perhaps we can conduct missions in this century out to 1,000 astronomical units where we might image a habitable exoplanet using a solar gravity lens as our telescope. If there is life out there we might actually see and touch it.

With such exploration there will be no need or motivation—cultural, scientific, or psychological—to have humans physically go beyond Mars. But humans will not be limited. At the same time we reach to the stars we will be settling on Mars, providing an infinite vista for growth and development. Occupying another world also addresses real or imagined existential threats to our home planet. Our species could thus survive even an asteroid impact, pandemic, climate catastrophe, or nuclear war. We will be settlers on Mars and explorers of worlds beyond, feeling our human presence throughout and beyond our solar system.

Back now to 2016 ... I see no contradiction or conflict in these two paths for human-robotic evolution—on Mars physically, in the Universe virtually. We are actually on these paths already, making me optimistic about our future as the most cosmic of explorers. □

Louis Friedman (louisdfriedman@gmail.com) co-founded The Planetary Society with Carl Sagan and Bruce Murray, serving as the organization's executive director for 30 years, worked on deep-space missions at the Jet Propulsion Laboratory in Pasadena, CA, and was co-leader of the Keck Institute for Space Studies Asteroid Retrieval Mission and Interstellar Medium Exploration Studies at Caltech.

© 2016 ACM 0001-0782/16/05 \$15.00

^a See the Committee's 2009 report at https://www.nasa.gov/pdf/396093main_HSF_Cmte_FinalReport.pdf and President Obama's announced goal in 2010 incorporating much of its recommendations at http://www.nasa.gov/news/media/trans/obama_ksc_trans.html.

From the intersection of computational science and technological speculation, with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/2904352

Louis Friedman

Future Tense Becoming a Multi-Planet Species

Imagine hyper-realistic virtual space exploration limited only by the data we are able to collect.

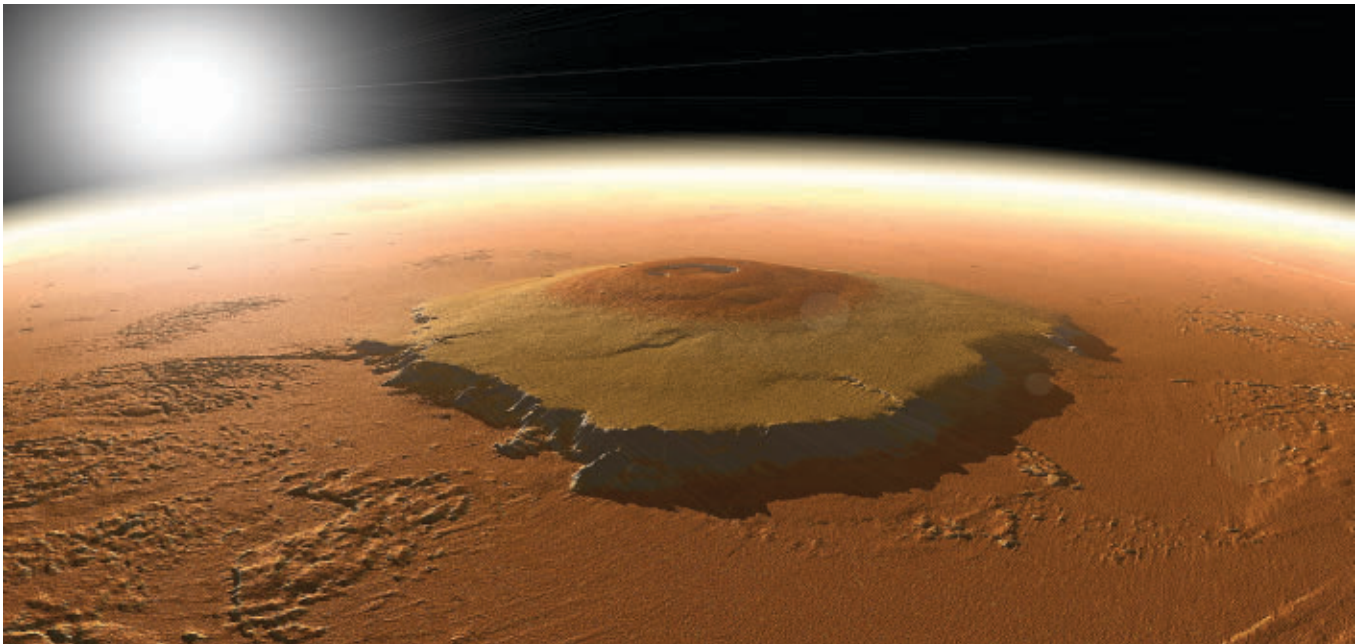
HUMANS AND ROBOTS together are symbiotically creating a new age of exploration, extending human presence and evolution to transform us into a multi-planet species. Not as humans vs. robots or as robots vs. humans. Such characterizations are archaic. Our multi-planet evolution will be a continuum of human and robot, with us physically present on Earth and Mars and virtually present beyond. Human-kind must make it to Mars to overcome the physical and psychological limit of being confined to our fragile singular planet. But technologies for exploration beyond Mars are evolving so quickly that by the time we settle on the Red Planet our species will be completely

immersed in and interacting with worlds throughout the solar system and even beyond. Let us imagine how that might be, looking into our future.

It is July 20, 2069, the 100th anniversary of humanity's first landing on another world. Dr. Angela Okonjo is reviewing, one last time, her prepared speech for the Anniversary Program. She is director of Mars Base 1, the highest-ranking human resident on Mars, whose population is now several hundred people distributed on three bases around the planet. "How strange it is," she thought, "that I, an information sciences major at MIT, immersed deeply in the world of robotics engineering and artificial intelligence, should be the representative of human

exploration on another world." But she realized it was not strange at all; all exploration involves humans and robots, humans deeply involved in so-called robotic missions, and robotics hard at work on so-called human missions.

She would commemorate not just the Apollo human Moon landing but a century of extraterrestrial robotic exploration, extending human presence throughout (but not yet beyond) the solar system. Remarkably, the "flexible path" for humans to Mars laid out 60 years ago had been followed—the first human venture beyond (albeit barely) the Moon in 2025; the first human interplanetary flight three years later; the first [CONTINUED ON P. 135]



DATA NASA/JPL/GSFC. IMAGE OLIVIER DE GOURSAC



Computing Reviews

20th ANNUAL BEST OF COMPUTING

Best Reviews Notable Books & Articles

The 20th Annual Best of Computing presents Computing Reviews' best reviews and the most interesting books and articles of the year.

The Best Reviews highlight the excellent contributions of the reviewers. Based on the recommendations from our reviewers, category editors, editors in chief of journals, and others in the computing community, the Notable Books & Articles list brings together influential items published in computing.

April 2016 Online & Print



Association for
Computing Machinery

ThinkLoud

www.computingreviews.com



fuse **DIS 2016**

Designing Interactive Systems

June 4 – 8

Brisbane Australia

 dis2016.org

 bit.ly/dis16

 [@DIS2016](https://twitter.com/DIS2016)



Association for
Computing Machinery



Queensland University of Technology
Brisbane Australia