

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

04/2015 VOL.58 NO.04

Sketch-Thru-Plan: A Multimodal Interface for Command and Control



Who Builds a House
without Drawing
Blueprints?

Security Challenges
for Medical Devices

Convolution Engine

Molecular Moonshots

Charting the Future:
Scholarly Publishing in CS



ACM Books



MORGAN & CLAYPOOL
PUBLISHERS

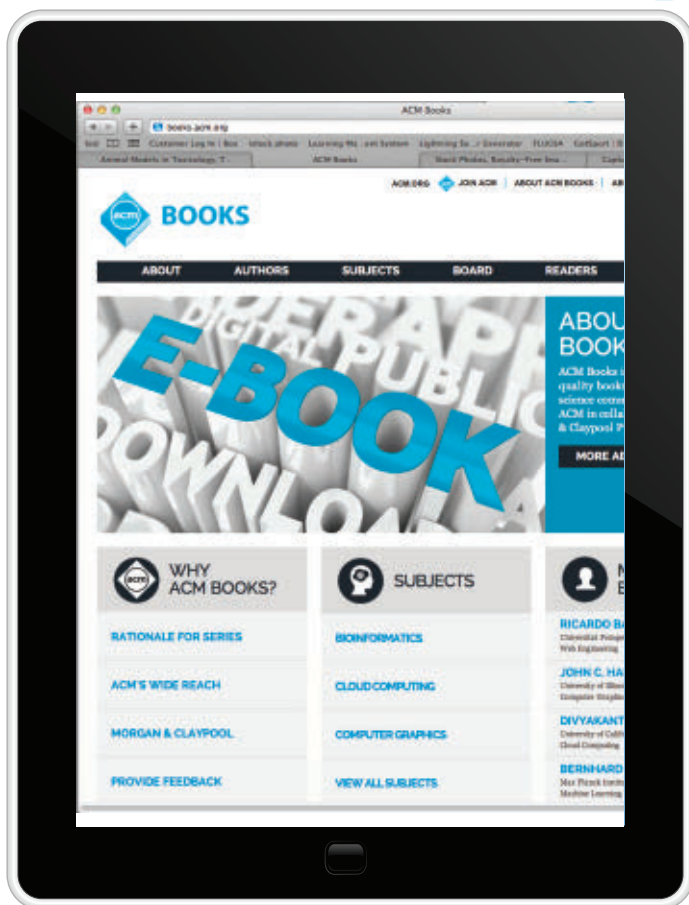
Publish your next book in the ACM Digital Library

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

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>**

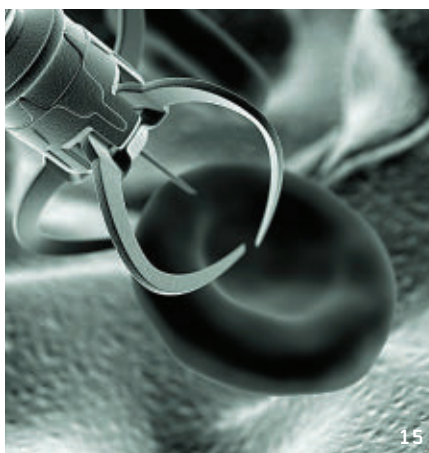
Departments

- 5 **From the Co-Chairs
of the ACM Publications Board**
**Charting the Future:
Scholarly Publishing in CS**
*By Joseph A. Konstan
and Jack W. Davidson*
-
- 7 **Cerf's Up**
The Human Touch
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Human or Machine?
-
- 12 **BLOG@CACM**
**The Arbitrariness of Reviews, and
Advice for School Administrators**
John Langford examines
the results of the NIPS experiment,
while Mark Guzdial considers
the role of class size in teaching
computer science.
-
- 39 **Calendar**
-
- 94 **Careers**

Last Byte

- 96 **Future Tense**
The Wealth of Planets
Launch swarms of self-replicating
robots to exploit the most lucrative
of resources.
By David Allen Batchelor

News



- 15 **Molecular Moonshots**
Synthetic biologists may be
closing in on potentially
world-changing breakthroughs,
but they are often hamstrung
by a shortage of software tools.
By Alex Wright
-
- 18 **Secure-System Designers Strive
to Stem Data Leaks**
Attackers using side-channel analysis
require little knowledge of how
an implementation operates.
By Chris Edwards
-
- 21 **What's the Price Now?**
Dynamic pricing finds its way into
a growing number of industries.
By Mark Broderick

Viewpoints

- 24 **Privacy and Security**
Toward More Secure Software
Two proposals intended
to reduce flaws in software use
two very different approaches
for software security.
By Dorothy E. Denning
-
- 27 **Technology Strategy and Management**
Competing in Emerging Markets
Considering the many different
paths and unprecedented
opportunities for companies
exploring emerging markets.
By Mari Sako
-
- 30 **Code Vicious**
Raw Networking
Relevance and repeatability.
By George V. Neville-Neil
-
- 33 **Interview**
An Interview with Juris Hartmanis
A pioneer in the field of
computational complexity theory
reflects on his career.
By Len Shustek
-
- 38 **Viewpoint**
**Who Builds a House without
Drawing Blueprints?**
Finding a better solution
by thinking about the problem
and its solution, rather than
just thinking about the code.
By Leslie Lamport



Practice



42

42 **Go Static or Go Home**

In the end, dynamic systems are simply less secure.

By Paul Vixie

46 **Hadoop Superlinear Scalability**

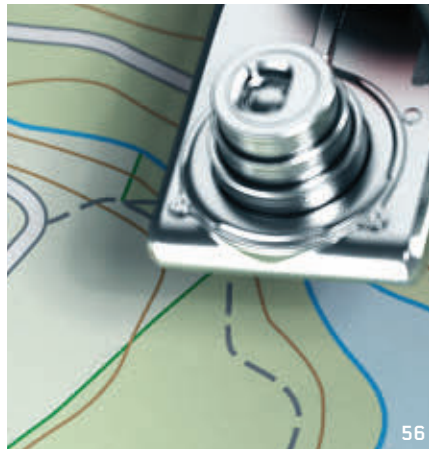
The perpetual motion of parallel performance.

By Neil J. Gunther, Paul Puglia, and Kristofer Tomasette



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

Contributed Articles



56

56 **Sketch-Thru-Plan:
A Multimodal Interface
for Command and Control**

Speaking military jargon, users can create labels and draw symbols to position objects on digitized maps.

By Philip R. Cohen, Edward C. Kaiser, M. Cecelia Buchanan, Scott Lind, Michael J. Corrigan, and R. Matthews Wesson



Watch the authors discuss this work in this exclusive *Communications* video.

66 **How Amazon Web Services
Uses Formal Methods**

Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

By Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, and Michael Deardeuff

Review Articles



74

74 **Security Challenges
for Medical Devices**

Implantable devices, often dependent on software, save countless lives. But how secure are they?

By Johannes Sametinger, Jerzy Rozenblit, Roman Lysecky, and Peter Ott

Research Highlights

84 **Technical Perspective
The Specialization Trend
in Computer Hardware**

By Trevor Mudge

85 **Convolution Engine:
Balancing Efficiency and Flexibility
in Specialized Computing**

By Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark Horowitz

**About the Cover:**

This month's cover story presents an advanced multimodal system called Sketch-Thru-Plan currently used for the rapid creation of plans for military ground operations. STP fuses map-based interfaces and mobile interfaces with alternative models of input, such as speech, touch, written actions, and more. Cover illustration by Justin Metz.



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

John White

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; Barbara Boucher Owens;

Andrew Sears

BOARD CHAIRS

Education Board

Mehran Sahami and Jane Chu Prey

Practitioners Board

George Neville-Neil

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Fabrizio Gagliardi

ACM India Council

Srinivas Padmanabhuni

ACM China Council

Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs

Jack Davidson; Joseph Konstan

Board Members

Ronald F. Boisvert; Nikil Dutt; Roch Guerrin;

Carol Hutchins; Yannis Ioannidis;

Catherine McGeoch; M. Tamer Ozsu;

Mary Lou Soffa

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

Lissa Clayborn, Acting 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

Public Relations Coordinator

Virginia Gold

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@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@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 Mehlforn;

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-Chairs

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; Bryan Cantrill;

Terry Coatta; Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Tom Limoncelli;

Kate Matsudaira; Marshall Kirk McKusick;

Erik Meijer; George Neville-Neil;

Theo Schlossnagle; Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of *queue*.

CONTRIBUTED ARTICLES

Co-Chairs

Al Aho and Andrew Chien

Board Members

William Aiello; Robert Austin; Elisa Bertino;

Gilles Brassard; Kim Bruce; Alan Bundy;

Peter Buneman; Peter Druschel;

Carlo Ghezzi; Carl Gutwin; 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; Amir El Abbadi; Sanjeev Arora;

Dan Boneh; Andrei Broder; Doug Burger;

Stuart K. Card; Jeff Chase; Jon Crowcroft;

Sandhya Dwaeekadas; Matt Dwyer;

Alon Halevy; Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Henry Kautz; Xavier Leroy;

Kobbi Nissim; Mendel Rosenblum;

David Salesin; Steve Seitz; Guy Steele, Jr.;

David Wagner; Margaret H. Wright

WEB

Chair

James Landay

Board Members

Marti Hearst; Jason I. Hong;

Jeff Johnson; Wendy E. MacKay

ACM Copyright Notice

Copyright © 2015 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 \$100.

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



Charting the Future: Scholarly Publishing in CS

ACM's Publications Board is charged with overseeing all of ACM's publishing activities, including publications policies, management of journals and magazines,

and the ACM Digital Library. We would like to update you on three key policy issues we are currently discussing, give you a glimpse of what some of the leaders of the field have told us, and invite your input into the discussion. The three issues are:

- ▶ the relationship between conference and journal publishing;
- ▶ the correctness-only reviewing model; and
- ▶ open access publishing models.

The relationship between conference and journal publishing is a perennial topic in computer science, in part because of our almost unique model of conference publishing in which top conferences have rigorous peer-review processes, high selectivity, and resulting high-impact proceedings. Different subfields within ACM vary, with some of them feeling top conferences are the preeminent publication venue, and others maintaining a distinction between the level of reviewing and quality of conference and journal papers. In many communities, including computer graphics and programming languages, there is pressure to publish top conference papers as papers in the top journals—in part because the status of conference publication is not universally recognized outside computer science and around the world.

The Publications Board has charged its Conferences Committee with studying this issue and making recommendations (the committee includes representatives of the conference-sponsoring SIGs). Our current policy recognizes two models for publishing conference papers in journals; a *journal-first* model where papers are submitted to and reviewed by the journal, but where accepted au-

thors (by a certain date) are invited to present at a conference, and a *journal-integrated* model where the conference review process has editorial oversight and feeds into the journal's process, permitting open-ended revisions and re-review by the same reviewers for papers that need revision. The principles behind these policies reflect the importance placed on a revision process that is not cut short by a conference event deadline.

To help inform our thinking, we surveyed ACM Fellows (receiving 166 responses as of this writing, a response rate between 20% and 25%). Fellows indicated they felt journals had higher-quality reviewing (67% to 14%) and higher-quality papers (46% to 25%), but that conference papers were more interesting (51% to 12%) and had higher impact on the field (57% to 21%). A large percentage (55% vs. 20%) indicated they prefer to publish in conferences, and 57% indicated they are more likely to use conferences publications to keep up on the field (vs. 14% for journals).

Correctness-only reviewing is an emerging publication trend in which journals commit to publishing all work they find to be technically sound without attempting to evaluate the importance or significance of the work (a successful journal built on this model is *PLOS ONE*). An argument for this model is that it speeds review and allows the community to see all the work being done and to judge its importance after publication. An argument against it is that readers want the review process to filter out work that is correct, but not significant (perhaps because it does not add much to the state of knowledge). As this model is being introduced into computer science (mostly by author-paid open access journals), we want-

ed to assess whether and how ACM should consider using the model.

Opinion among our surveyed ACM Fellows was divided. Some 41% thought correctness-only reviewing was bad for the field, while 36% felt there was a place for it. Only 11% felt ACM should integrate this model into our current publications, while 33% thought we should introduce a new line of clearly labeled correctness-only publications and 43% thought we should simply avoid correctness-only reviewing entirely.

Open access models are an area of broad interest, and we could fill a dozen columns on different issues related to open access publishing. Based on actions taken by certain research funders (primarily governmental, but also foundations), we have been looking at whether and how to incorporate author-pays open access into ACM's journals. We asked ACM Fellows about author-pays Gold OA journals, and specifically whether they preferred a Gold "umbrella" journal across computer science vs. Gold-only specialty journals vs. Gold editions of current journals. Gold-only specialty journals were preferred by 15% of Fellows; a Gold umbrella journal by 29%; and Gold editions of existing journals by 45%. Ten percent were against author-pays open access generally, preferring the current model or advocating for non-author-pays open access models.

What do you think? We would like your opinion. Please take the survey yourself at <https://www.surveymonkey.com/s/CACM-Pubs>; there are places for your free-form input as well. 

Joseph A. Konstan (konstan@umn.edu) and Jack W. Davidson (jwd@virginia.edu) are co-chairs of the ACM Publications Board.

Copyright held by authors.



HERE'S TO ADI, RON AND LEN.
FOR GIVING US RSA PUBLIC-KEY
CRYPTOGRAPHY.

We're more than computational theorists, database managers, UX mavens, coders and developers. We're on a mission to solve tomorrow. ACM gives us the resources, the access and the tools to invent the future. Join ACM today and receive 25% off your first year of membership.

BE CREATIVE. STAY CONNECTED. KEEP INVENTING.

[ACM.org/KeepInventing](https://acm.org/KeepInventing)



Association for
Computing Machinery



DOI:10.1145/2742906

Vinton G. Cerf

The Human Touch

The more I rely on software and programmed/programmable devices, the more I find I need help from a knowledgeable and timely reachable person. FAQs and YouTube videos

are often very helpful but when you get trapped into a situation where you have lost your ability to reach an online service, or your smartphone has become a useless brick, or your operating system has decided it hates you, you really want expert help fast. These things never happen at a convenient time. It's the holiday weekend, or 5 P.M. Friday, or 3 A.M. when you are pulling an all-nighter to get a big and important piece of work done. You do not want to hear "call us during office hours" or "we don't have a help service, see our FAQ."

As the global Internet allows vendors or equipment or online application providers to serve a global market, it becomes increasingly important to the globally distributed user population to be able to get assistance when it is needed. In some cases, self-service works very well. For example, overnight shippers often have a "where's my stuff" online help Web page that can tell you where your stuff was last seen (not where it *is*, necessarily). When you need contextual tutorial assistance, however, these methods may prove inadequate. Apple has made impressive strides with its Apple stores and "genius" assistants—and consumers seem to value consultation services enough to actually pay for them. One may have to schedule an appointment and appear in person, but for some situations, this may be the most effective way to treat such matters as configuration, equipment, or update problems.

Artificial intelligence may pave the way for some forms of local and timely help although I have my share of stories

with Siri and Google speech recognition systems, just for example, where they seem to misunderstand queries and commands deliberately. I recall one instance in which a map assistant persistently understood the address: "1312 Maple Street" as "131 to Maple Street" at which point it said it did not understand what I was asking. Recently, I was trying to find a restaurant in New York City, "Le Bernardin" and got "where is the burner diner in New York City?" No variation that I could find worked, including trying to spell the restaurant's name. Mispronouncing "lay Bernard inn" produced "The Barnard Inn" in Vermont. After a few more failures, you may imagine what I wanted to say to this otherwise helpful agent.

In many ways, I am optimistic that speech recognition will get better, especially as context and semantics are incorporated more deeply into the recognition systems. Yet there still seem to be a great many cases where human knowledge and understanding are vitally needed to solve customer problems. One thing that has become helpful is the ability to place phone calls or to open chat windows *in context* of a particular application. The ability to share the screen to show a customer service agent what you are seeing is also important. This morning, I ran into a problem with a payment system that required input in a particular field but refused to accept any of the choices it offered me to type in. I was stuck and so was the live chat room customer service representative who was unable to reproduce the re-

sults. A few hours later, I received an email message saying this was a new service and a bug had been found that caused the recalcitrant program to reject anything I entered for that field. It was gratifying to receive the feedback in a timely way.

As configuration complexity and scale grow, the need for smarter configuration systems, better online assistance, and the ability to share context with customer service agents will become increasingly important. It also seems as if the ability to export state information for later analysis or to inform customer service representatives of the current user's state will become valuable. However, this kind of access may also raise privacy and/or security concerns. Plainly, the computer science community has a lot to contribute, if it can figure out how to overcome these challenges.

While speaking of challenges and the human touch, it seems equally timely and relevant mention to the upcoming 25th Anniversary of the Americans with Disabilities Act on July 26, 2015. Making software that adapts to user needs is a big challenge but it is a vital objective. There are almost certainly laws like this in other countries where an effort has been or is being made to fashion adaptable user interfaces and experience to meet the assistive needs of many users of products and software services. 

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

Copyright held by author.

Human or Machine?

WE WISH to clarify an account of the 2014 Turing Test experiment we conducted at the Royal Society London, U.K., as outlined by Moshe Y. Vardi in his Editor's Letter "Would Turing Have Passed the Turing Test?" (Sept. 2014). Vardi was referring to a *New Yorker* blog by Gary Marcus, rather than to our experiment directly. But Marcus had no first-hand experience with our 2014 experiment nor has he seen any of our Turing Test conversations.

Our experiment involved 30 human judges, 30 hidden humans, and five machines—Cleverbot, Elbot, Eugene Goostman, JFred, and Ultra Hal; for background and details see <http://turingtest-sin2014.blogspot.co.uk/2014/06/eugene-goostman-machine-convinced-3333.html>. We used social media to recruit judges and a variety of hidden humans, including males, females, adults, teenagers, experts in computer science and robotics, and non-experts, including journalists, lecturers, students, and interested members of the public.

Prior to the tests, the judges were unaware of the nature of the pairs of hidden entities they would be interrogating; we told them only that they would simultaneously interrogate one human and one machine for five minutes and that the human could be a male or female, child or adult, native English speaker, or non-native English speaker. We asked the hidden humans to be themselves, that is, to be human.

The 30 judges, each given an anonymous experiment identity—labeled J1–J30—interrogated five pairs of hidden entities. Likewise each human and machine was given a unique identity—E1–E35. We ran 150 “simultaneous comparison” Turing Tests in which we instructed the judges that their task was to determine which was human and which was machine in the pair, a decision to be made based solely on the responses the hidden entities posted in reply to what a judge said.

Eugene Goostman was not correctly identified as the machine in the pair in 10 of its 30 tests; that is, 10 judges did not recognize it was a machine. Eugene

Goostman's personality is that of a 13-year-old boy from Odessa, Ukraine, a character we do not consider contrary to Alan M. Turing's vision for building a machine to think. In 1950, Turing said, “Instead of trying to produce a programme to simulate the adult mind, why not rather try to produce one which simulates the child's?”

The figure here includes one simultaneous conversation from the experiment, showing one of Judge J19's tests after that judge simultaneously interacted with two hidden entities, in this case E20 and E24. In this test, E20's responses to the judge were relayed to a message box displayed on the left of the judge's screen; E24's answers were

relayed on the right. Timings and text are exactly as they were in the test.

So, could you “pass the test” and be able to say which of the two entities—E20 and E24—is the human and which the machine?

Huma Shah, London, U.K., and

Kevin Warwick, Reading, U.K.

Author's Response:

The details of this 2014 Turing Test experiment only reinforces my judgment that the Turing Test says little about machine intelligence. The ability to generate a human-like dialogue is at best an extremely narrow slice of intelligence.

Moshe Y. Vardi, Editor-in-Chief

Simultaneous comparison by judge J19 in session four, round one of hidden entities E20 and E24 in a Turing Test.

Left	Right
[11:59:14] J19: Hello	[11:59:57] J19: How are you doing?
[11:59:18] E20: I was hoping to see you Judge!	[12:00:07] E24: I'm doing fine, you?
[11:59:25] J19: good!	[12:00:17] J19: great
[11:59:29] E20: That's good to hear.	[12:00:27] E24: That's good. :)
[11:59:38] J19: how are you	[12:01:07] J19: are you enjoying this process?
[11:59:41] E20: Everything's fine, Caused any scandals lately?	[12:01:17] E24: So far seems good.
[12:00:09] J19: no not really!	[12:02:02] J19: sorry if I am slow
[12:00:22] E20: Can you use better grammar and rephrase your question? I'd like to help, but I can't even make it through the jumbled mess. Use capitals and punctuation.	[12:02:12] E24: That's ok.
[12:00:34] J19: sorry	[12:02:31] J19: do you have a favorite tv show?
[12:00:38] E20: You're excused.	[12:02:47] E24: I have quite a few.
[12:00:49] J19: 't type as good as I talk	[12:03:01] J19: me too
[12:00:55] E20: Talk to me whenever you can.	[12:03:16] E24: What type of TV do you like?
[12:01:49] J19: Are you interested in artificial intelligence?	[12:03:34] J19: Game of thrones
[12:01:55] E20: I believe machines should do many to assist the lives of man.	[12:03:50] E24: I prefer documentaries.
[12:02:12] J19: I agree	
[12:02:17] E20: You are being well behaved today.	
[12:02:49] J19: as well behaved as I can be!	
[12:03:02] E20: Well then our peanut distracted me from the movie. Bad baby. [I look at your stomach as I say it, laughing] I don't mind if -.	
[12:03:19] J19: you're funny!	
[12:03:23] E20: Do you really think so? I'll always believe you're brilliant!	
[12:04:00] J19: thankyou remote19 left	
[12:04:03] E20: Don't think anything of it!	

Bahrain Revokes Masaud Jahromi's Citizenship

I have written several letters to the editor (June 2012, Jan. 2012, and Nov. 2011) about a citizen of Bahrain, professor and chair Masaud Jahromi of the Telecommunications Engineering Department at the Ahlia University in Bahrain, whose human rights had been violated by his own government. Jahromi was arrested and imprisoned in April 2011 for nearly six months for attending a rally on behalf of freedom. He was eventually tried, convicted, and sentenced by a court to five months in prison and a fine of approximately \$1,400. As he had already served five months, the court simultaneously suspended the four months. Following this January 19, 2012 ruling, he was dismissed from his position as professor and chair at Ahlia University only to be reinstated as professor February 20, 2012 and then as chair in March or April 2012.

The Bahrain Ministry of Interior has now revoked Jahromi's citizenship through a decree issued January 31, 2015. Jahromi was one of 72 Bahrainis, including journalists, activists, and doctors, to be stripped of their citizenship pursuant to a revision of the 1963 Bahrain Citizenship Act.

The Ministry of Interior announced its decree without court process or opportunity to respond, saying it was revoking the citizenship of the named individuals for "terrorist activities," including "advocating regime change through illegal means." There is no evidence Jahromi has ever participated in terrorism in any form. His sentence in 2011 was for participating in "unauthorized rallies" during protests. There are no additional allegations or evidence that he violated any law since then.

The summary revocation of citizenship appears to be a result of nonviolent expressive activity that has already been punished and not recurred. International instruments, including the Universal Declaration of Human Rights and the International Convention on Civil and Political Rights, to which Bahrain is a signatory, explicitly protect both the right of individuals to be free from arbitrary deprivation of their nationality and their right to

engage in nonviolent expressive activity. Indeed, Article 15 of the Universal Declaration of Human Rights specifically prohibits arbitrary deprivation of anyone's nationality.

Denial of citizenship without explanation or apparent basis imposes severe damage on an individual who consequently becomes stateless. Moreover, just including Jahromi's name on a list with the names of obvious terrorists serving with ISIS abroad damages Jahromi's reputation as an academic.

Jahromi believes wide publicity of his plight through *Communications* and support from the ACM membership was a positive factor in addressing his previous legal problem. Those who wish to help may write to the following address to request immediate restoration of Jahromi's Bahraini citizenship.

His Majesty Shaikh Hamad bin Issa Al Khalifa
King of Bahrain
Office of His Majesty the King
P.O. Box 555, Rifa'a Palace
Isa Town Central,
Kingdom of Bahrain
Jack Minker, College Park, MD

The Case of the Missing Skillset

In their article "Verifying Computations without Reexecuting Them" (Feb. 2015), Michael Walfish and Andrew J. Blumberg reviewed the state of the art in program verification while questioning whether to trust anything stored or computed on third-party servers, as in cloud computing, where companies and consumers alike access remote resources, including data, processing power, and memory, on a rental basis. Walfish and Blumberg proposed the formalism of probabilistically checkable proofs to allow, at least theoretically, a verifier to verify remotely performed computations. Not discussed, however, was an important aspect of verifiable software for the cloud—the general lack of the skillsets needed to write efficient and verifiable parallel programs.

As anyone in the software industry or related academic institutions can attest, training software engineers to use new tools and paradigms can involve an arduous learning curve.

Even if able to program in a new language, a developer might still need further training to become a productive member of a team. It takes time and resources to graduate from "Hello World" to programs fulfilling client use cases. Such an investment is perhaps what motivates organizations to outsource their software projects, a practice that risks even failure due to poor-quality software.²

Though developing verifiable software is not typically high on an organization's must-teach list, hosting an application in a cloud adds further requirements due to the code's remote service-driven execution architecture. As long as the cloud delivers the service at the promised quality of service, clients are unlikely to be interested in the details of its implementation. For example, Jeremy Avigad and John Harrison concluded in their article "Formally Verified Mathematics" (Apr. 2014), "There is a steep learning curve to the use of formal methods, and verifying even straightforward and intuitively clear inferences can be time consuming and difficult."¹

Yet another aspect of the problem is the absence of the skillset needed to write efficient cloud-based code, as not all code is readily convertible for parallel and cloud-friendly environments. While ample processing power may be available in a rental cloud, it may be difficult to find and train software developers to produce related high-quality cloud-based code.

Formalisms (such as those discussed by Walfish and Blumberg) may be exciting, at least theoretically, but realizing efficient and verifiable cloud code requires bridging the gap between the software industry and the community of formal-methods practitioners.

Muaz A. Niazi, Islamabad, Pakistan

References

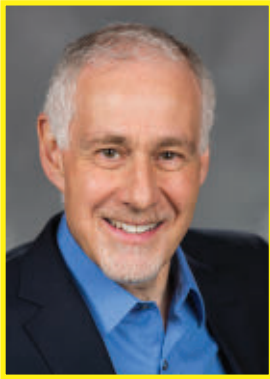
1. Avigad, J. and Harrison, J. Formally verified mathematics. *Commun. ACM* 57, 4 (Apr. 2014), 66–75.
2. Moe, N.B., Šmite, D., Hanssen, G.K., and Barney, H. From offshore outsourcing to insourcing and partnerships: Four failed outsourcing attempts. *Empirical Software Engineering* 19, 5 (Aug. 2014), 1225–1258.

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.

© 2015 ACM 0001-0782/15/04 \$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, appearing to read 'Alex Wolf'.

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

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:10.1145/2732417

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

The Arbitrariness of Reviews, and Advice for School Administrators

John Langford examines the results of the NIPS experiment, while Mark Guzdial considers the role of class size in teaching computer science.



John Langford
“The NIPS Experiment”

<http://bit.ly/1wZeUDK>
January 8, 2015

Corinna Cortes (<http://bit.ly/18I9RTK>) and Neil Lawrence (<http://bit.ly/1zy3Hjs>) ran the NIPS experiment (<http://bit.ly/1HNbXRT>), in which one-tenth of papers submitted to the Neural Information Processing Systems Foundation (NIPS, <http://nips.cc/>) went through the NIPS review process twice, and the accept/reject decision was compared. This was a great experiment, so kudos to NIPS for being willing to do it and to Corinna and Neil for doing it.

The 26% disagreement rate presented at the NIPS conference (<http://bit.ly/18Iaj4r>) understates the meaning in my opinion, given the 22% acceptance rate. The immediate implication is that one-half to two-thirds of papers accepted at NIPS would have been rejected if reviewed a second time. For analysis details and discussion about that, see <http://bit.ly/1uRCqCF>.

Let us give P (reject in 2nd review | accept 1st review) a name: arbitrariness. For NIPS 2014, arbitrariness was ~60%. Given such a stark number, the primary question is “what does it mean?”

Does it mean there is no signal in the accept/reject decision? Clearly not—a purely random decision would have arbitrariness of ~78%. It is, however, notable that 60% is closer to 78% than 0%.

Does it mean the NIPS accept/reject decision is unfair? Not necessarily. If a pure random number generator made the accept/reject decision, it would be ‘fair’ in the same sense that a lottery is fair, and have an arbitrariness of ~78%.

Does it mean the NIPS accept/reject decision could be unfair? The numbers make no judgment here. It is a natural fallacy to imagine random judgments derived from people imply unfairness, so I would encourage people to withhold judgment on this question for now.

Is arbitrariness of 0% the goal? Achieving 0% arbitrariness is easy: choose all papers with an md5sum that ends in 00 (in binary). Clearly,

there is more to be desired from a reviewing process.

Perhaps this means we should decrease the acceptance rate? Maybe, but this makes sense only if you believe arbitrariness is good, as it will almost surely increase the arbitrariness. In the extreme case where only one paper is accepted, the odds of it being rejected on re-review are near 100%.

Perhaps this means we should increase the acceptance rate? If all papers submitted were accepted, the arbitrariness would be 0, but as mentioned earlier, arbitrariness of 0 is not the goal.

Perhaps this means NIPS is a broad conference with substantial disagreement by reviewers (and attendees) about what is important? Maybe. This seems plausible to me, given anecdotal personal experience. Perhaps small, highly focused conferences have a smaller arbitrariness?

Perhaps this means researchers submit to an arbitrary process for historical reasons? The arbitrariness is clear, the reason less so. A mostly arbitrary review process may be helpful in that it gives authors a painful-but-useful opportunity to debug easy ways to misinterpret their work. It may also be helpful in that it rejects the bottom 20% of papers that are actively wrong, and hence harmful to the process of developing knowledge. These reasons are not confirmed, of course.

Is it possible to do better? I believe the answer is “yes,” but it should be understood as a fundamentally difficult problem. Every program chair who cares tries to tweak the reviewing process to be better, and there have been

many smart program chairs that tried hard. Why is it not better? There are strong nonvisible constraints on the reviewers' time and attention.

What does it mean? In the end, I think it means two things of real importance.

1. The result of the process is mostly arbitrary. As an author, I found rejects of good papers hard to swallow, especially when reviews were nonsensical. Learning to accept the process has a strong element of arbitrariness helped me deal with that. Now there is proof, so new authors need not be so discouraged.

2. The Conference Management Toolkit (<http://bit.ly/16n3WCL>) has a tool to measure arbitrariness that can be used by other conferences. Joelle Pineau and I changed ICML 2012 (<http://bit.ly/1wZiZaW>) in various ways. Many of these appeared beneficial and some stuck, but others did not. In the long run, it is things that stick that matter. Being able to measure the review process in a more powerful way might be beneficial in getting good practices to stick.

You can see related commentary by Lance Fortnow (<http://bit.ly/1HNfPm7>), Bert Huang (<http://bit.ly/1DpGf6L>), and Yisong Yue (<http://bit.ly/1zNvoqb>).



Mark Guzdial
"Rising Enrollment
Might Capsize
Retention and
Diversity Efforts"

<http://bit.ly/1J3lst0>
 January 19, 2015

Computing educators have been working hard at figuring out how to make sure students succeed in computer science classes—with measurable success. The best paper award for the 2013 SIGCSE Symposium went to a paper on how a combination of pair programming, peer instruction, and curriculum change led to dramatic improvements in retention (<http://bit.ly/1EB9mle>). The chairs award for the 2013 ICER Conference went to a paper describing how Media Computation positively impacted retention in multiple institutions over a 10-year period (<http://bit.ly/1AkPH2x>). The best paper award at ITICSE 2014 was a meta-analysis of papers exploring approaches to lower failure rates in CS undergraduate classes (<http://bit.ly/1zNrvBH>).

How things have changed! Few CS departments in the U.S. are worried about

retention right now; instead, they are looking for ways to manage rising enrollment threatening to undermine efforts to increase diversity in CS education.

Enrollments in computer science are skyrocketing. Ed Lazowska and Eric Roberts sounded the alarm at the NCWIT summit last May, showing rising enrollments at several institutions (see <http://tern.ch/1zxUho2> and charts at right). Indiana University's undergraduate computing and informatics enrollment has tripled in the last seven years (<http://bit.ly/1EBaX0K>). At the Georgia Institute of Technology (Georgia Tech), our previous maximum number of undergraduates in computing was 1,539 set in 2001. As of fall 2014, we have 1,665 undergraduates.

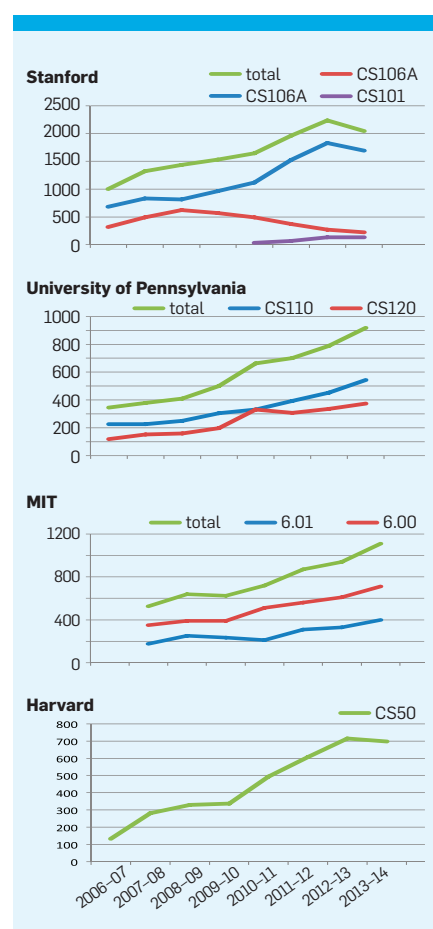
What do we do? One thing we might do is hire more faculty, and some schools are doing that. There were over 250 job ads for CS faculty in a recent month. I do not know if there are enough CS Ph.D.'s looking for jobs to meet this kind of growth in demand for our courses.

Many schools are putting the brakes on enrollment. Georgia Tech is limiting transfers into the CS major and minor. The University of Massachusetts at Amherst is implementing caps. The University of California at Berkeley has a minimum GPA requirement to transfer into computer science.

We have been down this road before. In the 1980s when enrollment spiked, a variety of mechanisms were put into place to limit enrollment (<http://bit.ly/1KkZ9hB>). If there were too few seats available in our classes, we wanted to save those for the "best and brightest." From that perspective, a minimum GPA requirement made sense. From a diversity perspective, it did not.

Even today, white and Asian males are more likely to have access to Advanced Placement Computer Science and other pre-college computing opportunities. Who is going to do better in the intro course: the female student trying out programming for the first time, or the male student who has already programmed in Java? Our efforts to increase the diversity of computing education are likely to be undermined by efforts to manage rising enrollment. *Students who get shut out by such limits are most often in demographic groups underrepresented in computer science.*

When swamped by overwhelming numbers, retention is not your first



priority. When you start allocating seats, students with prior experience look most deserving of the opportunity. Google is trying to help; it started a pilot program to offer grants to schools with innovative ideas to manage booming enrollment *without* sacrificing diversity (<http://bit.ly/16b292F>).

One reason to put more computer science into schools is the need for computing workers (<http://bit.ly/1DxYbN3>). What happens if kids get engaged by activities like the Hour of Code, then cannot get into undergraduate CS classes? The linkage between computing in schools and a well-prepared workforce will be broken. It is ironic our efforts to diversify computing may be getting broken by *too many* kids being sold on the value of computing!

It is great students see the value of computing. Now, we have to figure out how to meet the demand—without sacrificing our efforts to increase diversity. ▣

John Langford is a principal researcher at Microsoft Research New York. Mark Guzdial is a professor at the Georgia Institute of Technology.

© 2015 ACM 0001-0782/15/04 \$15.00



Distinguished Speakers Program

talks by and with technology leaders and innovators

Chapters • Colleges and Universities • Corporations • Agencies • Event Planners

A great speaker can make the difference between a good event and a WOW event!

The Association for Computing Machinery (ACM), the world's largest educational and scientific computing society, now provides colleges and universities, corporations, event and conference planners, and agencies – in addition to ACM local Chapters – with direct access to top technology leaders and innovators from nearly every sector of the computing industry.

Book the speaker for your next event through the ACM Distinguished Speakers Program (DSP) and deliver compelling and insightful content to your audience. **ACM will cover the cost of transportation for the speaker to travel to your event.** Our program features renowned thought leaders in academia, industry and government speaking about the most important topics in the computing and IT world today. Our booking process is simple and convenient. Please visit us at: www.dsp.acm.org. If you have questions, please send them to acmdsp@acm.org.

The ACM Distinguished Speakers Program is an excellent solution for:

Corporations Educate your technical staff, ramp up the knowledge of your team, and give your employees the opportunity to have their questions answered by experts in their field.

Colleges and Universities Expand the knowledge base of your students with exciting lectures and the chance to engage with a computing professional in their desired field of expertise.

Event and Conference Planners Use the ACM DSP to help find compelling speakers for your next conference and reduce your costs in the process.

ACM Local Chapters Boost attendance at your meetings with live talks by DSP speakers and keep your chapter members informed of the latest industry findings.

Captivating Speakers from Exceptional Companies, Colleges and Universities

DSP speakers represent a broad range of companies, colleges and universities, including:

AMD
Carnegie Mellon University
Google
IBM

Imperial College London
INTEL
Lawrence Berkeley Nat'l Laboratory
Microsoft

Nanyang Technological University
Raytheon BBN Technologies
Stanford University
Tsinghua University

UCLA
University of British Columbia
University of Cambridge
University of Texas at Austin

Topics for Every Interest

Over 500 lectures are available from more than 120 different speakers with topics covering:

Software
Cloud and Delivery Methods
Emerging Technologies
Engineering

Web Topics
Computer Systems
Open Source
Game Development

Career-Related Topics
Science and Computing
Artificial Intelligence
Mobile Computing

Computer Graphics, Visualization
and Interactive Techniques
High Performance Computing
Human Computer Interaction

Exceptional Quality Is Our Standard

The same ACM you know from our world-class Digital Library, magazines and journals is now putting the affordable and flexible Distinguished Speaker Program within reach of the computing community.

Microsoft®
Research
The DSP is sponsored
in part by Microsoft Europe



**Association for
Computing Machinery**

Advancing Computing as a Science & Profession

Molecular Moonshots

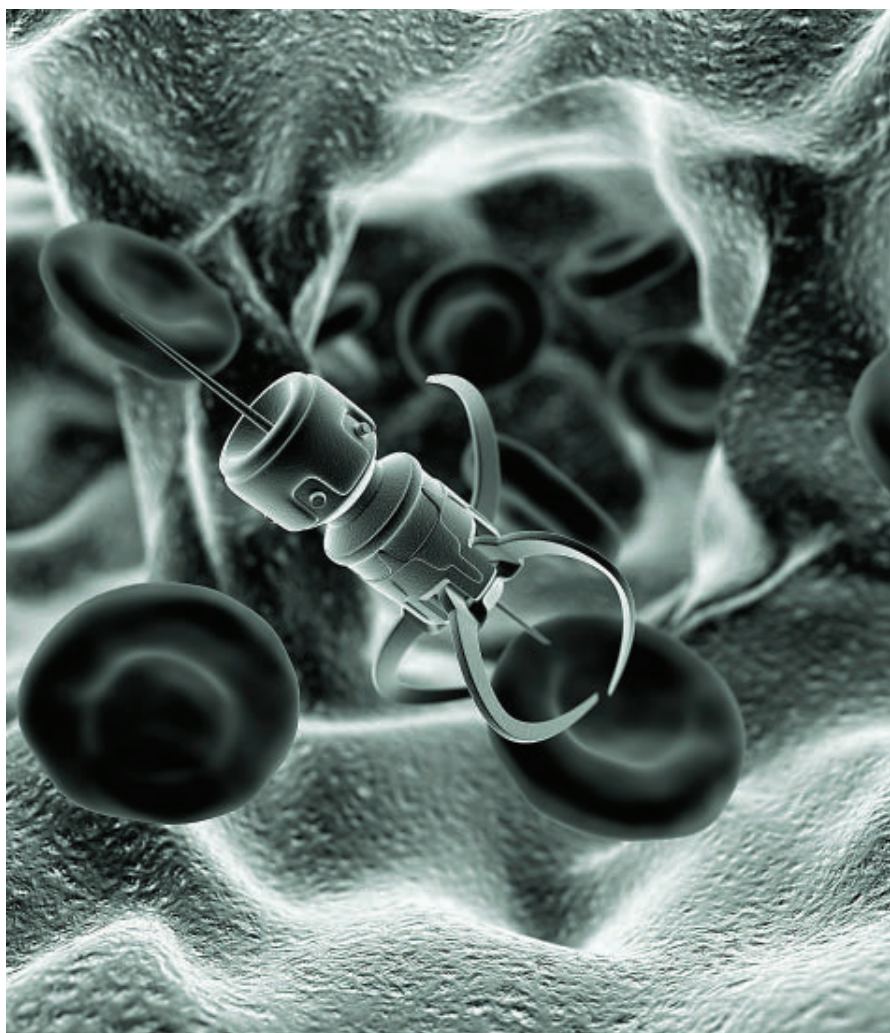
Synthetic biologists may be closing in on potentially world-changing breakthroughs, but they are often hamstrung by a shortage of software tools.

WHEN A TEAM of researchers at Bar-Ilan University in Israel recently announced they had successfully implanted DNA-based nanorobots inside living cockroaches—possibly paving the way for a revolution in cancer treatment—it marked the latest in a series of promising innovations to emerge from the synthetic biology community over the past decade.

In recent years, biotechnologists have started to come tantalizingly close to engineering next-generation drugs and vaccines, DNA-based computational systems, and even brand-new synthetic life forms. Amid all these advances, however, the development of synthetic biology software has largely failed to keep up with the pace of innovation in the field.

With only a handful of commercial software tools at their disposal, most synthetic biologists have had no choice but to build their own bespoke systems to support the intense data modeling needs of molecular engineering.

“Computer science is critically important to synthetic biology,” says Drew Endy, an associate professor of biotechnology at Stanford University and co-founder of BIOFAB, an ambitious



effort to create open source biological parts for molecular engineering. Yet surprisingly few computer scientists have chosen to enter the field.

That situation may be slowly changing, as a handful of developers have started to delve into the challenges of molecular engineering. One of the most promising initiatives to date has come from Autodesk, which is developing a software platform designed for synthetic biology, 3D bioprinting, 4D printing, and DNA nanotechnology, code-named Project Cyborg.

"DNA is the universal programming language," says Andrew Hessel of Auto-desk's Bio/Nano/Programmable Matter group, who sees enormous potential in applying the principles of computer science to biological applications. "The architectures of biology—from proteins to metabolism to cells to tissues to organisms and ecosystems—it's all just layered systems on systems. But you need the foundation of silicon computing to support

that work. Otherwise, DNA is just as unreadable as ones and zeros."

Project Cyborg offers researchers and designers a set of Web-based computer-aided design (CAD) tools that allow the engineering of nano-scale objects in a visual design framework. The system allows users to store and manipulate that data in the cloud. Designs can be fabricated by a number of DNA "print shops" around the world (such as DNA2.0 in Menlo Park or Gen9 in Boston).

Thinking about the design of living things, Hessel has discovered irresistible parallels between synthetic biology and computing. "Cells are just living, squishy parallel processors," says Hessel. "They are fully programmable."

One of Project Cyborg's best-known beta testers is Shawn Douglas at the University of California San Francisco, who is using the Autodesk tools to design cancer-fighting nanorobots that have already proven effective at isolating and attacking

cancer cells in a lab environment.

The nanorobots work by twisting into a precise configuration that allows them to attach to targeted cells, looking for signals from antigens on the cells' surface that flag them as cancerous. The nanorobots then lock onto the cell, breaking it open and engaging the cell receptors to trigger a response that destroys the cancer cell.

Transforming DNA strands into predetermined shapes requires a complex and delicate process of manipulating nucleotides to assemble in a pre-ordained sequencing, precisely ordering the genomic components of adenine, cytosine, guanine, and thymine. It is painstaking work, involving thousands of detailed specifications.

Douglas initially used the open source drawing tool Inkblot (and in some cases, pen and paper). Frustrated with the limitations of the available software, he drew on his own background in computer science to develop an open-source software tool

Milestones

Celebrations Honor Boole's 200th Birthday

University College Cork (UCC), which describes itself on its website as "the academic home of George Boole," is preparing a number of events in celebration of the 200th birthday of Boole, the English logician, mathematician, and philosopher who introduced what has become known as Boolean algebra and Boolean logic.

Born in Lincoln, England, in 1815, Boole became a schoolteacher and researched mathematics in his spare time. In 1844, he was awarded the Gold Medal of the Royal Society in London for his paper "On a General Method in Analysis." He was named UCC's first professor of mathematics in 1849, and is often said to have laid the foundations of the Information Age, as his work serves as the basis for modern mathematics, microelectronic engineering, and computer science.

Last November, the university launched a year-long celebration of Boole's bicentenary, to include high-profile projects and events commemorating his contributions in the fields of mathematics, computer science, and engineering. In addition,

a series of legacy initiatives are planned, both to inspire and to fund the next generation of talent in those disciplines.

Speaking at the launch of the commemorative year, Ireland's Taoiseach (prime minister) Enda Kenny said Boole's "remarkably beautiful work is integral to cybernetics, logic, information theory, computer science; every time we turn on a laptop or take a call on our mobile phone or a cold drink from the fridge, we can thank George Boole."

"He was extraordinary, not alone in the brilliance and fineness of his mind but in the depth of his heart. His compassion for his fellow human beings, his sense of responsibility toward them, saw him make a significant contribution both to adult education and the learning and welfare of the poor and the excluded."

Highlights of the George Boole 200 celebration include:

- ▶ Restoration of No.5 Grenville Place, Boole's first home in Cork.
- ▶ An official film biography of George Boole.
- ▶ An interactive tour "led" by George Boole, featuring key

locations on the UCC campus where Boole taught, is being provided to the public.

▶ The exhibition *Boolean Expressions: Contemporary Art and Mathematical Data* will begin July 30 and run through Nov. 8 at UCC's Lewis Glucksman Gallery.

▶ The George Boole 200 Inaugural Lectures kicked off in February with Muffy Calder, professor of computing science at the University of Glasgow, and Alberto Sangiovanni Vincentelli, professor of electrical engineering and computer science at the University of California, Berkeley, addressing Boole's legacy. For more information on those lectures, visit: <http://www.georgeboole.com/events/lectures/george-boole-200-inaugural-lectures.html>.

▶ UCC will host a series of conferences to celebrate Boole's work and legacy. The Mathematical Sciences Conference is scheduled for Aug. 17; the Boole Bicentenary Celebration of his achievements and multiple legacies is scheduled for Aug. 28–30; the 21st International Conference on Principles and Practice of

Constraint Programming and the 31st International Conference on Logic Programming are both scheduled for Aug. 31–Sept. 4, and a conference on engineering is slated for some time in September.

▶ A series of outreach events to primary school children is planned to celebrate Boole and demonstrate his relevance to today's younger generation. One of these is the 'I Wish' event, led by Cork Chamber of Commerce and Cork City Council, which will give 1,000 female transition-year students the chance to meet female entrepreneurs and other female role models from science, technology, engineering, and mathematics (STEM) fields.

▶ On the 200th anniversary of Boole's birth on Nov. 2, a celebration at UCC will include the conferring of honorary degrees on leaders in the fields of mathematics and information technology from around the world.

▶ Other events at UCC and elsewhere will be added to George Boole 200 during the course of the year. For the latest news on George Boole 200, visit www.georgeboole.com.

—Lawrence M. Fisher

known as Cadnano to accelerate the process.

The work paid off, and soon researchers in his lab were able to complete dozens of designs per month, when previously they were only completing one. They soon began to take on more challenging design projects.

One project originated with Douglas' work on DNA as a graduate student at Harvard University in the mid-2000s, working with collaborators Hendrik Dietz and William Shish to create complex shapes made out of DNA strands. By manipulating the order of nucleotides within the famous double-helix structure of DNA, they were able to influence the way one strand interacted with another: in effect, programming them to bend, join together, and combine into new shapes.

Douglas continued to pursue his studies at Harvard with computational biology pioneer George M. Church and Ido Bachelet, a researcher in experimental pharmaceuticals who would go on to lead the work on DNA-based origami robots at Bar-Ilan University. Eventually, they began to work on progressively more ambitious DNA experiments.

In 2012, Douglas and Bachelet published a seminal article in *Science* about their work, chronicling the results of their experiments in designing an autonomous DNA nanorobot capable of delivering molecular "payloads" to attack six distinct types of cancer cells. Their study found this approach worked particularly well with certain kinds of leukemia cells, while leaving 99% of healthy cells untouched.

If successful, nanorobot-based cancer treatment would represent a huge leap forward from the brute force therapies of traditional chemotherapy and radiation techniques, and promises to relieve the suffering and prolong the lives of countless millions of cancer patients.

Elsewhere, researchers at Duke University, the University of Rome, and Aarhus University have built a "DNA nanocage" with tiny lattices that can open and close in response to temperature signals. This mechanism would allow it to release enzymes in response to environmental conditions, making it theoretically possible to deliver highly targeted medicines in pill form.

For all of their potentially enormous promise, none of these devices has

"Cells are living, squishy processors," says Hessel. "They are fully programmable."

been tested on humans. The potential side-effects and complications remain unclear. Unlike insects, human beings and other mammals will be more difficult to penetrate due to their higher levels of infection-fighting nucleases, but researchers are continuing to work on solutions to that problem.

That work will require increasingly complex computer modeling, almost certainly involving more custom application development. Indeed, most of the high-profile synthetic biology work—and funding—has focused on the "application" layer of high-profile solutions, rather than on the far-less-glamorous systems layer that would ultimately enable more and more of these applications to take shape.

"Historically, the engineering toolkit for biotechnology has not been well-defined," says Endy, who laments the lack of funding support—and apparent interest—in tackling these infrastructure-layer challenges in both the academic or corporate worlds. None of the major university computer science departments devote significant support to synthetic biology, nor (apart from Autodesk) does there seem to be much interest emanating from the corporate software world.

"Teams that get funded to work on solving problems with biology tend to be overdriven by the immediate pressing nature of the situation," says Endy. "That leads to a bias in funding and underdevelopment. We are too focused on the applications."

Endy feels the answer may lie in public-private partnerships, similar to the ones he forged with BIOFAB, which pioneered the development of an open source framework known as BIOBRIC, something akin to a Creative Commons license for biological parts.

"Public-private partnerships can work in biotechnology by allowing aca-

demics and professionals to work together in an effective way," says Endy. "Academics are in the business of shipping papers, and biotech companies are over-driven by implementation." Endy sees potentially enormous complementary power at work when the corporate work ethic gets coupled with the academic commitment to getting the fundamentals right.

Moreover, biologists and computer scientists still need to learn how to speak each other's languages. DNA may be a programming language, but it does not involve a binary system; instead, it is an ordinal system, involving a more complex code of nucleotide sequences. And while most computer programming languages are linear, time-based systems, biology happens in the physical world; it's a four-dimensional proposition.

Taking computer science from the binary realm of silicon into the four-dimensional, ordinal biological world will inevitably require a steep learning curve, but the potential payoff for humanity is enormous. Says Endy: "The number of miracle moonshots is almost infinite." **C**

Further Reading

Amir, Y., Ben-Ishay, E., Levner, D., Ittah, S., Abu-Horowitz, A., Bachelet, I. **Universal Computing by DNA origami robots in a living animal.** *Nature Nanotechnology*. 9, 353-357 (2014). doi:10.1038/nnano.2014.58.

Douglas, S.M., Bachelet, I., Church, G.M. **A Logic-Gated Nanorobot for Targeted Transport of Molecular Payloads.** *Science*. 335:831-4 (2012).

Mutalik, V., Guimaraes, J., Cambray, G., Lam, C., Christoffersen, M.J., Mai, Q., Tran, A.B., Paull, M., Keasling, J.D., Arkin, A.P., Endy, D. **Precise and reliable gene expression via standard transcription and translation initiation elements.** *Nature Methods*. 10: 354-360 (2013). doi:10.1038/nmeth.2404

Sissel, J., Iacovelli, F., Falconi, M., Kragh, S., Christensen, B., Fröhlich, R., Franch, O., Kristoffersen, E., Stougaard, M., Leong, K.W., Ho, Y., Sørensen, E.S., Birkedal, V., Desideri, A., Knudsen, B.R.

Temperature-Controlled Encapsulation and Release of an Active Enzyme in the Cavity of a Self-Assembled DNA Nanocage. *ACS Nano*, 2013, DOI: 10.1021/nn4030543

Alex Wright is a writer and information architect based in Brooklyn, NY.

© 2015 ACM 0001-0782/15/04 \$15.00

Secure-System Designers Strive to Stem Data Leaks

Attackers using side-channel analysis require little knowledge of how an implementation operates.

CHIP AND SYSTEM designers are engaged in a cat-and-mouse battle against hackers to try to prevent information leaking from their circuits that can be used to reveal supposedly secure cryptographic keys and other secrets.

Traditionally, such side-channel attacks have relied on expensive bench-top instruments such as digital-storage oscilloscopes, but the development of an open-source platform dubbed ChipWhisperer based on affordable programmable integrated circuits (ICs) has widened the potential user base, as well as making it easier for designers to assess the vulnerability of their own designs. The latest addition to the ChipWhisperer platform developed by Colin O'Flynn, a doctoral student at Dalhousie University in Nova Scotia, Canada, and colleague Zhizhang Chen is based on a \$90 board that is able to recover secret keys from simple microcontroller-based targets in a matter of minutes, although it is by no means an automated process.

"Because the way in which the attack works, it's very important to understand the theory behind it," O'Flynn points out. However, an important feature of this class of attack is that it focuses on the core algorithm, rather than on the idiosyncrasies of a particular implementation.

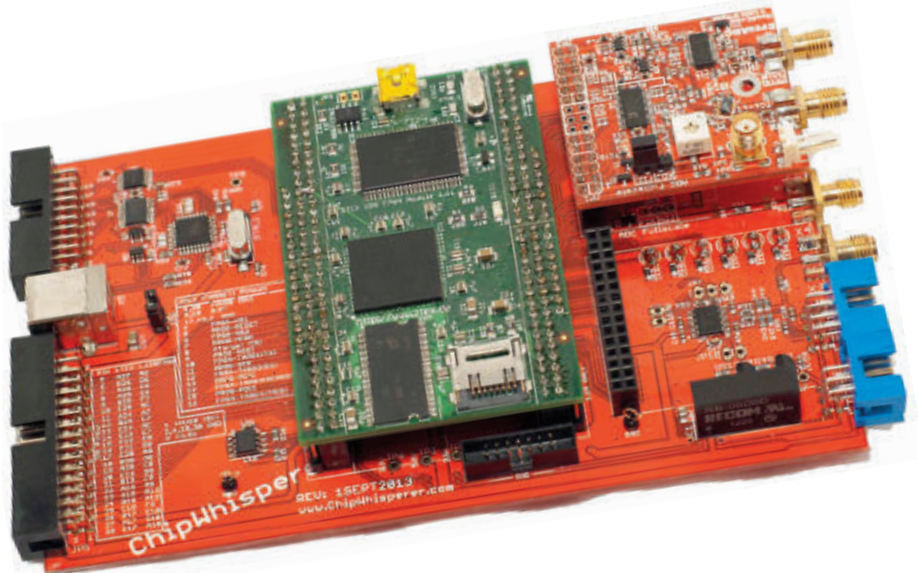
To Patrick Schaumont, associate professor of electrical and computer engineering and the director of the Center for Embedded Systems for Critical Applications at Virginia Polytechnic Institute and State University (Virginia Tech), "What makes side-channel analysis so impressive and so scary is that people who have access to your implementation need to make very few assumptions on what

is actually happening inside. They are able to work toward success based purely on statistics."

The key to side-channel analysis is that changes in data as they are processed by algorithms running on a microprocessor or dedicated hardware unit yield a detectable fingerprint, which may be picked up as changes in the power consumed by the target or as heat or electromagnetic emissions. Daniel Mayer, senior applications

the operation of a circuit in research by leading cryptologist Adi Shamir, based at the Weizmann Institute of Science and working with colleagues from Tel Aviv University. Changes in the amount of current passing through the transformer cause oscillations that can be heard as subtle changes in sound.

Timing-based attacks provide the basis for some of the simplest forms of side-channel analysis. O'Flynn cites a now-discontinued hard drive that



A board built on the ChipWhisperer platform can recover secret keys from microcontroller-based targets in minutes.

security consultant at New York City-based security research firm Matasano Security, explained at the recent Black Hat USA conference: "They are all related to the computation that the system does at a given time. Using that information, picked up outside the application, you can infer something about the secret it contains."

Even the sounds from the windings of a transformer in a power supply have been used to collect information about

used a PIN code entered on its panel to provide access to users. An attacker could measure how long the drive's firmware took to analyze different codes by simply iterating through integers at each position in the six-digit PIN. "As soon as the while-loop to check the PIN fails, it just exits. There should be a million combinations of this password. But even in the worst case for this drive, it takes just 60 tries to guess the PIN."

Measuring timing makes it possible to perform side-channel attacks across a network instead of requiring that the target equipment be in the hands of the hacker. Some network services tend to be highly vulnerable. “Web API keys are places where developers commonly perform comparisons insecurely,” says Joel Sandin, a security consultant at Matasano Security who worked with Mayer on side-channel analysis research.

The Matasano researchers and others have confirmed that statistical analysis can help the attacker to discern data-dependent changes in execution time, although actual exploitation is not straightforward as there is no direct way to separate application execution time from network latency. “With a remote-timing attack, we can’t measure execution time directly; we can only measure round-trip time,” says Sandin.

Algorithms can be written to avoid giving away information easily, but power and electromagnetic emissions provide more subtle clues about the data being processed, even if the application designer takes precautions. Some attacks focus on the differences in the complexity of logic blocks used to perform cryptographic operations.

A naïve implementation of the modular exponentiation operations common to many cryptographic algorithms calls for a squaring operation when the relevant key bit is 0, and a square and a multiply when the key bit is 1. A binary multiply operation generally involves many more logic steps than the squaring operation, which can be implemented as a simple shift of the data in the word to the left. As a result, the multiply’s power consumption should be higher.

Typically, the attacker will, as with the timing attacks, provide known text to the target system. As the text changes, the attacker can record the shifts in power consumption caused by different logic blocks being activated. The power and activity difference between the shift and multiply operations makes the analysis relatively simple, and has successfully uncovered keys on the low-end microcontrollers used in early smartcards using less than 100 different pieces of source plaintext.

As designers have implemented techniques to hide the differences

Algorithms can be written to avoid giving away information easily, but power and electromagnetic emissions provide more subtle clues about the data being processed.

between operations, sometimes by performing fake multiplications, attention in side-channel analysis has focused on the data itself. O’Flynn says the Hamming weight—the number of logic ones in a binary word—provides important clues about the state of the target as data moves through it.

“Inside a digital device there will be different modules. In between them there will be a data bus. It takes physical electrons to change logic levels, which takes power. If, at one instant in time, it takes more power than at another, it is probably setting more bits high,” O’Flynn explains, and that difference in power can be described using the Hamming weight.

The connection between power and Hamming weight is linear. Some algorithms, such as the S-box part of the AES algorithm, have a more obvious, non-linear relationship between source data and result. This, says O’Flynn, provides potentially richer data for side-channel analysis if designers do not find ways to disguise what the algorithms are doing internally.

Countermeasures against side-channel attacks mainly focus on ways to disguise the data being leaked, increasing the number of operations the attacker needs to collect for a successful breach. One key technique is masking, in which a random value is combined with real data on the way into the cryptoprocessor. The use of random values obscures the Ham-

ACM Member News

CIRCUITOUS PATH LEADS ACKERMAN TO COMPUTER SCIENCE



Mark Ackerman’s path to computer science was a circuitous one.

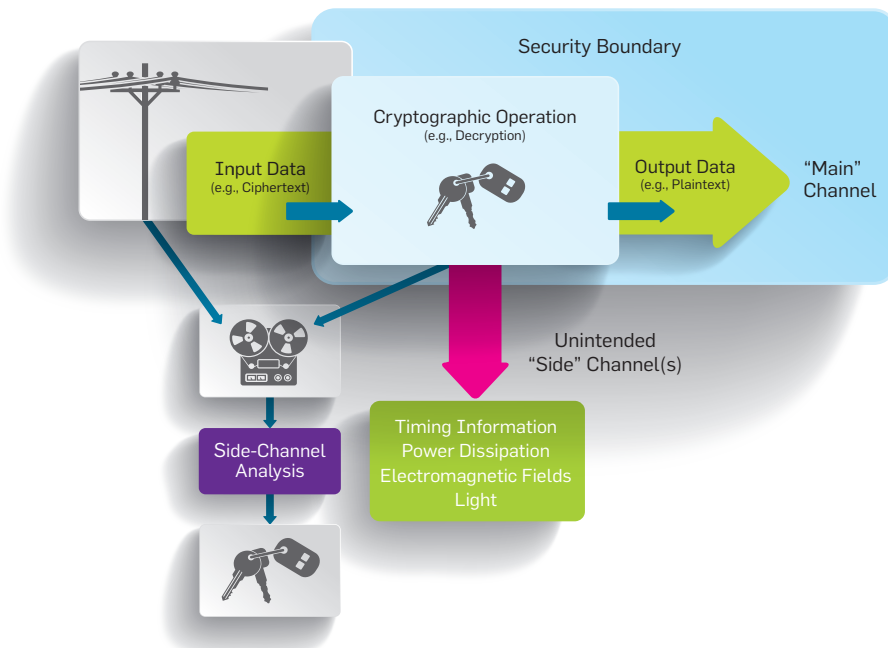
Now a professor in the Department of Electrical Engineering and Computer Science and in the School of Information at the University of Michigan, Ann Arbor, Ackerman recalls, “I started out as a history major, considered and rejected being a lawyer, switched to computer science and have no regrets.”

Along the way, Ackerman took creative career detours: he led the development of the nation’s first home banking system; designed three Billboard Top-10 games for General Computer’s Atari 2600, and worked on the X Window System’s first user-interface widget set.

Ackerman earned a B.A. in history from the University of Chicago in 1979, an M.S. in computer and information science from Ohio State University in 1981, and a Ph.D. in information technologies with a secondary concentration in Media Technology from the Massachusetts Institute of Technology in 1994. His present research focus is Human-Computer Interaction (HCI) and Computer-Supported Cooperative Work. “I always wanted to find ways of interacting with computers socially,” Ackerman says.

Ackerman’s current HCI research focuses on collective memory in the Internet of Things (IoT) age. “HCI is living history; some devices will remember stories,” he says. He also is working on collaborative applications involving IoT and social implications. “Privacy issues are the flip side of information access,” Ackerman observes. “There are GPS devices installed in cars for people with high-risk loans; you can get discounts on auto insurance if you let the insurance companies track you. Will these capabilities be invasive, or will they yield useful knowledge?”

—Laura DiDio



Secret information leaks out of the security boundary via unintended side channels.

ming weights of successive pieces of data, reducing the correlation between the predictions made by the attacker's model of the algorithm and the results of the cryptographic operations. The effect of this operation is reversed after processing, to recover the properly encrypted data.

To try to give designers a better understanding of the effectiveness of countermeasures, researchers have tried to model them analytically. A team led by Virginia Tech's Schaumont has developed the first of a series of algorithms that calculate at the source-code level the resistance to attacks that different masking schemes can achieve. Mayer and Sandin demonstrated a tool at Black Hat that helps analyze the vulnerability of software running on a server to timing-based attacks. In other work, Yensu Fei and colleagues from Northeastern University developed a statistical model to calculate the degree to which activity from other modules in the target chip can mask the emissions that could be used by an attacker.

Researchers maintain countermeasures cannot render crypto circuitry immune to side-channel attacks. Says Chris Woods, researcher at Quo Vadis Labs and a member of a team from the University of Cambridge in the U.K. that developed techniques for optimizing such attacks, "Side-channel attacks can break any countermeasure, given

enough time; the countermeasure can only delay the process. Any chip will leak information; it's just a matter of trying to mitigate the leakage."

Hackers have found profitable targets that make the breaking of countermeasures worthwhile. Embedded systems such as set-top boxes can provide rich pickings for pirates if they are able to crack master keys in such a way that enables them to provide access codes to users who pay for them. Some hackers are prepared to invest in sophisticated hardware and manpower to improve their chances of success of recovering keys.

Some devices contain counters and other circuitry to detect anomalous use—wiping the secret keys if the at-

"Side-channel attacks can break any countermeasure, given enough time; the countermeasure can only delay the process."

tacker exceeds a threshold value, which is much more likely if masking is used in the core algorithm. Yet even here, the hackers have tools they can use to prevent the counters being employed. Targeted electrical glitches can prevent these protection circuits from operating correctly. More sophisticated attacks use lasers fired at a key part of the chip surface, taking advantage of the sensitivity to some wavelengths of light that silicon transistors exhibit.

Peter Ateshian, a researcher at Dutch security specialist Riscure and at the Naval Postgraduate School in Monterey, CA, says, "Pirates will spend up to \$10 million or so for cracking a new system-on-chip or new set-top box element. They will build their own devices to see how to extract information from the video stream."

Governments can find reasons to invest even more in breaking the encryption on a device they have captured or confiscated. "If we are talking about national governments, they have effectively unlimited resources," Ateshian says.

Woods warns if the attacker's model of the system's behavior is good enough, "you will be able to break the device no matter what." **C**

Further Reading

Kocher, P.C., Jaffe, J., Jun, B. *Differential Power Analysis, Proceedings of the 19th Annual Conference on Advances in Cryptology (CRYPTO '99)* <http://bit.ly/1z0DaKr>

O'Flynn, C., Chen, Z. *ChipWhisperer: An Open-Source Platform for Hardware Embedded Security Research, Constructive Side-Channel Analysis and Secure Design Lecture Notes in Computer Science (2014)* <https://eprint.iacr.org/2014/204.pdf>

Eldib, H., Wang, C., Taha, M., Schaumont, P. *QMS: Evaluating the Side-Channel Resistance of Masked Software from Source Code, Proceedings of the 51st Annual Design Automation Conference (2014)* <http://dl.acm.org/citation.cfm?id=2593193>

Fei, Y., Ding, A.A., Lao, J., Zhang, L. *A Statistics-based Fundamental Model for Side-channel Attack Analysis, IACR Cryptology ePrint Archive 2014, 152 (2014)* <https://eprint.iacr.org/2014/152>

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

© 2015 ACM 0001-0782/15/04 \$15.00

What's the Price Now?

Dynamic pricing finds its way into a growing number of industries.

IT IS FAIRLY common for travelers to begin their searches for the ideal itinerary online, only to find the price of a plane ticket or hotel room has changed dramatically in just a few hours. Checking back on one of the many travel comparison websites a few days later will only result in more exasperation, as a hotel room price may have gone up, while the cost of the airline ticket has gone down.

These industries rely on pricing models that try to eke the most profit out of a plane ticket or hotel room. The constantly changing selling price is by design: travel providers have decades of experience in using dynamic pricing models based on supply and demand, which help them balance prices with seat and room availability. Yet there is far more to dynamic pricing than basic economics, and a number of industries—including some unexpected ones—are adopting this concept to capitalize on this growing trend of price optimization.

What is really enabling the growth of dynamic pricing is the availability of petabytes of data stemming from the billions of transactions that consumers have conducted with businesses on a daily basis. Coupled with advances in software, inexpensive storage solutions, and computers capable of analyzing thousands of variables, this vast amount of data can be crunched almost instantly to help companies optimize pricing on everything from a pair of shoes to a luxury automobile.

There are dozens of software firms offering intelligent pricing software, many designed with specific vertical markets in mind. At the center of each vendor's offering is their own proprietary algorithm used to help the deployer maximize profits. The algorithms are an integral part of the software, taking in all of the data and making pricing recommendations in near-real time. The companies deploying the software also can customize the weight of different inputs to help achieve their

What is enabling the growth of dynamic pricing is the availability of petabytes of data from billions of consumer transactions.

desired outcome. Online retailers that want to ensure pricing for “hot” items is in line with competitors may choose to adjust prices more frequently than on slower-moving products.

Airlines in particular have become adept at using virtual warehouses of data to help set pricing at the seat level. The result: passengers sitting in the same row on a plane, just inches from one another, may have paid drastically

different prices for their seats. Supply and demand are certainly key factors that determine the cost of a ticket, but competitors' prices, seasonality, the cost of fuel, and a host of other variables go into the computation of prices, all based on complex algorithms.

While airlines were the pioneers in the use of dynamic pricing science, the retail industry has emerged as a leader in innovation and implementation of price optimization systems.

According to a November 2013 *Retail Info Systems News* survey, 22.6% of online retailers are utilizing pricing intelligence software to assist in price optimization. An additional 35.6% of survey respondents said they expected to implement such systems within the next year. If these retailers followed through with their implementation plans, that means more than half of online retailers are using pricing intelligence systems today.

Arnoud Victor den Boer of the University of Twente's Stochastic Operations Research group says, “The emergence of the Internet as a sales channel has made it very easy for companies to



experiment with selling prices.” In the past, companies had to replace price tags on shelves or print new catalogs when prices changed, but online retailers have much more flexibility, and can instantly change prices based on market dynamics. “This flexibility in pricing is one of the main drivers for dynamic pricing,” says den Boer.

The ‘Amazon Effect’

What might be called an ‘Amazon effect’ is prompting retailers across many vertical markets to take a closer look at price optimization strategies. Pricing intelligence software can scan competitors’ websites as often as every 10 minutes for their pricing and, using pre-set rules, automatically raise or lower prices on products to stay competitive.

Den Boer says online retailers are leading the pack in the use of dynamic pricing, “but a growing number of companies in ‘offline’ settings are starting to acknowledge the possible advantages of dynamic pricing. The idea that your customer has a ‘personal’ willingness to pay which you would like to learn/exploit is very appealing.”

Dynamic pricing is being used in areas that might, at least on the surface, seem to be incompatible with this technology. Such models are being used to determine the cost of the cars we drive, the amount we pay for parking for that

car and, perhaps one day, how much we spend on fueling the vehicle.

One of the most complex uses of dynamic pricing may be in the retail automobile market. A car is generally one of the largest purchases a consumer will make in their lifetime, second only the purchase of a home. Santa Monica, CA-based TrueCar Inc. has developed a proprietary methodology designed to help the three primary constituents in the car-buying process—the manufacturer, the dealer, and the consumer—complete a deal that is equitable for all parties.

John Williams, senior vice president of technology at TrueCar, says, “Dynamic pricing within automotive retail has been an ever-present market force; however, it is in many ways more complex and more foundational to the process than in other industries. It is often the case that two different shoppers can walk into the same store, on the same day, buy the exact same car, and pay completely different prices—sometimes as much as a 30% difference.” The same pricing phenomenon may also be observed in other markets, but is amplified in a car purchase, where the price tag is in the tens of thousands of dollars.

According to Williams, “What’s especially fascinating about this inherent price variability in automotive retail is that it generally is inefficient for both

the buyer and seller.” TrueCar surveyed car buyers, asking how much profit they thought a dealer makes; consumers on average believed the average margin on a car is 19.7%. When asked what a fair profit margin would be, car buyers said 13.2%; in reality, the average dealer profit margin is closer to 3%–4%. “This shows that there is a huge opportunity to produce better outcomes for both buyers and sellers within the marketplace,” says Williams.

The volume of data TrueCar must crunch is huge, requiring parallel computing and an enormous amount of storage. The company compares buyers’ needs with dealers’ existing inventories, adding in variables for what other buyers have paid, vehicle condition, demand for specific cars, and hundreds of other data elements. The result is a complex algorithm sitting on top of a massive data warehouse. TrueCar’s Williams said, “We built a multi-petabyte Hadoop cluster for \$0.29/GB, a truly game-changing price point. Storing all the ambient data within a marketplace provides a powerful historical record to use for many dynamic pricing applications, including techniques like machine learning. Historical data is used to train machine learning models; the more history you can feed it, the smarter the algorithms are, and that yields a marketplace advantage.”

Milestones

Computer Science Awards, Appointments

MICROSOFT AWARDS RESEARCH FACULTY FELLOWSHIPS

Microsoft Research recently awarded Microsoft Research Faculty Fellowships to seven promising early-career academics engaged in innovative computing research who have the potential to make significant advances in the state of the art.

These fellowships include a cash award and access to software, invitations to conferences, and engagements with Microsoft Research.

The new Microsoft Research Faculty Fellows are:

► Vinod Vaikuntanathan, assistant professor of computer science at the Massachusetts

Institute of Technology, who played a prominent role in the development of lattice-based cryptography and leakage-resilient cryptography, and has focused his research on the theory and practice of computing on encrypted data.

► David Steurer, assistant professor of computer science at Cornell University, who investigates the power and limitations of efficient algorithms for optimization problems in computer science and its applications.

► Roxana Geambasu, assistant professor of computer science, Columbia University, who works at the intersection of distributed systems, operating systems, and security and privacy.

► Yong-Yeol Ahn, assistant professor of informatics and computer science at Indiana University Bloomington, whose research uses mathematical and computational methods to study complex systems.

► Percy Liang, assistant professor of computer science at Stanford University, whose research interests include parsing natural language into semantic representations for supporting intelligent user interfaces, and developing machine learning algorithms that infer rich latent structures from limited supervision, balancing computational and statistical trade-offs.

► Byung-Gon Chun, assistant

professor of computer science at Seoul National University, who is developing a big data platform that will make it easy to implement large-scale, fault-tolerant, heterogeneous data processing applications.

► Diego Fernández Slezak, assistant professor of computer science at the School of Exact and Natural Sciences, University of Buenos Aires, whose work focuses on novel methods for text analysis in massive-scale repositories to find stereotyped patterns in human thought, with the goal of developing machine-learning techniques to study digital text corpora associated with cognitive processes.

New distributed computing solutions are changing both how data is stored and processed. Hadoop includes a highly scalable file system called HDFS that distributes data across many high-performance servers. Because those servers also have powerful CPUs and plenty of RAM, data crunching is handled by distributed programs that run on many cooperating servers at once. Techniques such as Map Reduce, originated by Google, are used to analyze huge amounts of data on Hadoop clusters that can contain thousands of servers. TrueCar uses this approach to model the automotive marketplace, creating giant lookup tables that contain every possible outcome to the many variables within the ecosystem. These lookup tables are then pushed into front-end databases and provide lightning-fast “real time” results when users begin adjusting options and exploring different possibilities. The next step in the evolution of these systems is real-time streaming of data and continuous data processing across the Hadoop cluster using technologies such as YARN and Storm, which manage resources for even faster data processing.

TrueCar has revolutionized the car-buying process; more than 2% of all vehicles sold in the U.S. are now through TrueCar.

Parking

Another unexpected market segment utilizing dynamic pricing is the parking industry. Up until a few years ago, companies that operated street parking essentially counted the number of coins in a meter to determine demand for parking; not only was this inefficient, but the operators also had no idea when parkers were coming or going, or how long they were parked. Municipalities worldwide have adopted electronic meters capable of not only accepting credit cards, but also able to set pricing based on space availability and time of day. Sensor-equipped meters are connected to a hub that can tell how many spaces are available on a street and adjust pricing accordingly. The goal is not to gouge consumers, but rather to optimize the price so there is always a space available—if the customer is willing to pay the price.

An important factor in dynamic pricing is the ability to develop “decent, well-performing scientific algorithms to determine sales prices based on accumulating sales data.”

Periods of lower demand mean lower prices, but when demand is high, customers should expect to pay more.

In the retail petroleum market, there is already a significant amount of variability in the price of gasoline at the pump, generally based on the price the gas station has negotiated with its supplier. This market has yet to embrace dynamic pricing at the pump, but it could be an area where we see some development. A customer pumping 20 gallons of gas, for example, might be offered preferred pricing when compared to another customer that only pumps five gallons. Discounts are already offered at some gas stations for customers that pay using cash rather than credit cards, and the next area of development in this market could take into account additional variables including the number of gallons pumped, the time of day, or even the weather (offering discounts on snowy days when demand might ordinarily be lower).

Start-Ups

Start-ups look at dynamic pricing as a relatively untapped market. Even in the travel industry, where dynamic pricing models have been in use since the 1980s, there are significant opportunities. At the November 2014 PhoCusWright Conference, a venture capital pitch-style event for travel companies, a number of presenters' business ideas were built upon dynamic pricing models. Options Away, for example, was the first company to

present at the conference, offering a unique take on airline ticket purchasing. The company's product enables potential travelers to purchase options on ever-fluctuating plane tickets, locking in prices while they firm up travel plans. The company has already successfully partnered with leading ticket providers.

There is no lack of innovative ideas based on dynamic pricing. The success of dynamic pricing can be highly reliant on all of the variables in a given market. According to den Boer, success may depend on the ability to understand customer segments and customers' willingness to pay different prices, as well as the ability to collect and interpret historical data. Another important factor is the ability to develop “decent, well-performing scientific algorithms to determine sales prices based on accumulating sales data.”

The horizon for this growing field is wide open. There are already applications in a far-ranging number of fields, from credit card issuers determining customers' interest rates based on their credit scores to the price riders pay for a car ride on Uber during peak hours. In the coming years, developers will continue to work on new and innovative ideas using dynamic pricing models, and it could alter the dynamics of an even broader range of industries. ■

Further Reading

Den Boer, A.V.

Dynamic Pricing and Learning: Historical Origins, Current Research, and New Directions
University of Twente, 11 December 2014
<http://bit.ly/1GjwaIY>

Le Guen, T.

Data-Driven Pricing
Massachusetts Institute of Technology, 2008
<http://bit.ly/1wgvHoH>

Robles, P.

Five Dynamic Pricing Issues Retailers Should Consider
Econsultancy, 25 January 2013
<http://bit.ly/1vEoiLZ>

Proffitt, B.

How Much Will It Cost You? With Dynamic Pricing, Online Sellers Say “It Depends”
ReadWrite.com, 6 September 2012
<http://bit.ly/1GjBlSu>

Mark Broderick is a Tampa, FL-based senior research analyst covering the financial services and payments industries for ORC International.

© 2015 ACM 0001-0782/15/04 \$15.00

Privacy and Security

Toward More Secure Software

Two proposals intended to reduce flaws in software use two very different approaches for software security.

LAST YEAR, THE National Institute Standards and Technology (NIST) added 7,937 vulnerabilities to the National Vulnerability Database (NVD), up from 5,174 in 2013. That is approximately 22 per day, or almost one every hour. Of these, 1,912 (24%) were labeled “high severity” and 7,243 (91%) “high” or “medium.”⁷ Simply put, they cannot be ignored.

As I read reports of new vulnerabilities and the risks they enable, I wonder whether it will ever end. Will our software products ever be sufficiently secure that reports such as these are few and far between? Or, will they only become more prevalent as more and more software enters the market, and more dangerous as software increasingly controls network-enabled medical devices and other products that perform life-critical functions?

In this column, I will explore two proposals aimed at reducing software flaws. The first, which involves the U.S. government cornering the vulnerability market, I believe, could make the problem worse. However, the second,

which involves holding companies liable for faulty software, is an idea whose time has come.

Cornering the Vulnerability Market

Many software companies, including Microsoft, Google, and Mozilla, operate bug bounty programs, paying security researchers who bring new security flaws to their attention. Other companies serve as brokers, buying vulnerabilities and exploits from security researchers, and then selling or donating them to product vendors and other customers. To the extent the end consumers in this growing market are the companies whose products are flawed, the market serves to strengthen software security. But when end consumers are intelligence agencies and criminals who use the information to exploit target systems, the vulnerability market exposes us all to greater risk.

To further reduce software vulnerabilities beyond what the market has achieved so far, we could look for ways that encourage the pursuit of vulnerabilities with the goal of getting them fixed, but discourage their sale and use

for target exploitation. One possibility, suggested by Dan Geer, is for the U.S. government (USG) to openly corner the vulnerability market.⁶ In particular, the USG would buy all vulnerabilities and share them with vendors and the public, offering to pay say 10 times as much as any competitor. Geer argues this strategy will enlarge the talent pool of vulnerability finders, while also devaluing the vulnerabilities themselves. Assuming the supply of vulnerabilities is relatively sparse, the approach could eventually lead to a situation where most vulnerabilities have been exposed and fixed, rendering any cyber weapons that exploited them useless. In addition, since researchers finding new zero-day will maximize their earnings by selling them to the USG, fewer zero-days should end up in the hands of adversaries.

The cost of Geer’s proposal seems reasonable. Current prices for vulnerabilities range from a few hundred to several hundred thousand dollars. If we consider the approximately 8,000 vulnerabilities added to the NVD in 2014 and assume an average price of \$1,000, then the cost of purchas-



```
public final class Test {

    private static final String BUG = "bug";

    public static boolean bug(String s) {
        return "bug".equals(s.isEmpty() ? BUG : s);
    }

    private Test() {}
}

<plugin>
  <groupId>org.codehaus.mojo</groupId>
  <artifactId>findbugs-maven-plugin</artifactId>
  <version>3.0.0</version>
</plugin>
```

than in patching flaws in deployed code that puts users at risk?

Another issue is whether the USG would be willing to disclose all vulnerabilities. Under current policy, software flaws it uncovers are generally to be disclosed to vendors in order that they can be patched, but they can be kept secret and exploited when there is “a clear national security or law enforcement” reason.¹⁰ At the same time, it seems likely that many discovered vulnerabilities will never reach the USG, being held for the purposes of exploitation by criminals and foreign governments. And many persons might simply oppose reporting them to the USG.

Finally, vulnerability disclosure has the downside of increasing the risks of those using the reported products, at least until they can acquire and install the necessary patches. Consider ShellShock, a flaw in the UNIX Bourne-again shell (Bash), which lets attackers remotely execute code or escalate privileges. Disclosure of the flaw allowed attackers to harvest vulnerable computers into a botnet that sent out over 100,000 phishing email messages.³ A Symantec study found that attacks exploiting particular zero-day vulnerabilities increased by as much as 100,000-fold after their disclosure.²

Software Liability

A better approach to reducing vulnerabilities would be to hold software companies liable for damages incurred by cyber-attacks that exploit security flaws in their code. Right now, companies are not liable, protected by their licensing agreements. No other industry enjoys such dispensation. The manufacturers of automobiles, appliances, and other products can be sued for faulty products that lead to death and injury.

In *Geekonomics*, David Rice makes a strong case that industry incentives to produce secure software are inadequate under current market forces, and that one way of shifting this would be to hold companies accountable for faulty products.⁸ Geer proposes that software companies be liable for damages caused by commercial software when used normally, but that developers could avoid liability by delivering their software with “complete and buildable source code and a license that allows disabling any functional-

ing these vulnerabilities would be \$8 million, a drop in the bucket for the USG. Even if the average price rose to \$100,000, the annual cost would still be reasonable at \$800 million. However, the costs could become much higher and the problems worse if the program perversely incentivized the creation of bugs (for example, an inside developer colluding with an outside bounty collector).¹ Costs could also rise from outrageous monetary demands or the effects of more people looking for bugs in more products.

I especially worry that by shifting

the cost from the private sector to the USG, companies would lose an economic incentive to produce more secure software in the first place. As it is, an empirical study by UC Berkeley researchers of the bug bounty programs offered by Google and Mozilla for their respective browsers, Chrome and Firefox, found the programs were economically efficient and cost effective compared to hiring full-time security researchers to hunt down the flaws.⁵ Would it not be better to shift the incentives so it was more economical to invest in secure software development

ity or code the licensee decides.”⁶ The escape clause, which would cover free and open source software, would allow users to inspect and cut out any software they did not trust.

My main concern with Geer’s proposal relates to absolving any code offered commercially from liability. As a practical matter, very few users are in a position to inspect source code. Even those that are can miss significant flaws, as seen with Heartbleed, a flaw in OpenSSL that gives attackers access to sensitive information, and also ShellShock. In addition, exemption does nothing to incentivize the production of more secure open source code. At the same time, penalizing a large, volunteer community for flaws in their code would be both difficult and distasteful.

A better way around this dilemma might be to exempt the immediate developers of open source code, but hold accountable any company that embeds it in their products or that offers services for open source products. Under such a provision, if an individual or group of volunteers offers a free, open source App, they would not be accountable, though any company offering it through their App store would be.

This compromise would incentivize software companies to pay greater attention to security in all of the code they offer through their products and services, regardless of whether the code is developed in house, by a contractor, or within the open source community; and regardless of whether the product is released with source code or the capability to disable certain functions. Moreover, given that many companies contribute to open source development, they would be incentivized to promote secure coding practices in the open source community as well as within their own development teams.

Importantly, as with other forms of liability, software liability should be tied to standards and best practices, as well as the damage and harm that result from security flaws. The objective is not to penalize companies who invest considerable resources in software security but find their code vulnerable to a new exploit that nobody had anticipated. Rather, it is to bring all software up to a higher level of security by punishing those who are negligent in this domain, for example, by putting out code that

Developing a suitable liability regime will be a challenge, as the system must address the concerns of both software developers and users.

fails to check inputs. Standards and best practices for secure coding have advanced considerably, and readers interested in learning more might start with CERT’s Secure Coding Web portal.⁴

The argument is often made that software liability will inhibit innovation, but we should inhibit the introduction of faulty software. Moreover, assigning liability will likely stimulate innovation relating to secure software development. Another argument against software liability is that it will raise the price of software. While this may be true, it should lower the costs we all pay from cyber-attacks that exploit software vulnerabilities, costs that have been rising over the years and fall on the backs of users as well as software companies.

Conclusion

Bug bounties emerged under current market forces and are likely here to stay. I oppose a program that would attempt to have the U.S. government corner and fund this market, in part because it would reduce the incentive for software companies to produce more secure code and could make the problem worse.

A better strategy is one that increases the incentives for the development of secure software but decreases those for putting out sloppy code. One way of achieving this is by holding companies responsible for all the code they sell and service, including both proprietary and open source. Under this strategy, companies could be sued for damages caused by cyber-attacks that exploited flaws in their code, and penalties would be inflicted according to whether the code was developed under standards and best practices for secure coding.

Because software licenses and the Uniform Commercial Code severely limit vendors from liability for security flaws in their code, companies today cannot be effectively sued or punished when they are negligent and the flaws are exploited to cause economic harm.⁹ Legislation or regulation is needed to change this and remove the ability of companies to exempt themselves through licensing agreements. Developing a suitable liability regime will be a challenge, however, as the system must address the concerns of both software developers and users. Perhaps a good start would be for ACM to sponsor a workshop that brings together a diverse community of stakeholders and domain experts to recommend a course of action.

Of course, holding software companies accountable will not solve all our security woes. Many cyber-attacks exploit weaknesses unrelated to faulty software, such as weak and default passwords and failure to encrypt sensitive information. But companies are liable when their systems are attacked, and they can be successfully sued for not following security standards and best practices. The time has come to make software vendors liable as well. ■

References

1. Baker, H. Re: Zero-day bounties. *The Risks Digest* 28.25 (Sept. 9, 2014).
2. Bilge, L. and Dumitras, T. An empirical study of zero-day attacks in the real world. *CCS’12* (Oct. 16–18, Raleigh, N.C.); http://users.ece.cmu.edu/~tdumitra/public_documents/bilge12_zero_day.pdf.
3. Carman, A. Shellshock used to amass botnet and execute phishing campaign. *SC Magazine* (Oct. 15, 2014); <http://bit.ly/1Df7Slg>.
4. CERT Coding Standards; <http://www.cert.org/secure-coding/index.cfm>.
5. Finitzer, M., Akhawe, D., and Wagner, D. An empirical study of vulnerability rewards programs. *USENIX Security 13*; <https://www.eecs.berkeley.edu/~daw/papers/vrp-use13.pdf>.
6. Geer, D. Cybersecurity as realpolitik. *Blackhat 2014*; <http://geer.tinho.net/geer.blackhat.6viii14.txt>.
7. National Vulnerability Database, National Institute of Standards and Technology; <https://web.nvd.nist.gov>.
8. Rice, D. *Geekonomics: The Real Cost of Insecure Software*. Addison Wesley, 2008.
9. Scott, M.D. Tort liability for vendors of insecure software: Has the time finally come? *Maryland Law Review* 67, 2 (2008); <http://digitalcommons.law.umaryland.edu/cgi/viewcontent.cgi?article=3320&context=mlr>.
10. Zetter, K. Obama: NSA must reveal bugs like Heartbleed, unless they help the NSA. *Wired* (Apr. 15, 2014); <http://www.wired.com/2014/04/obama-zero-day/>.

Dorothy E. Denning (dedennin@nps.edu) is Distinguished Professor of Defense Analysis at the Naval Postgraduate School in Monterey, CA.

The views expressed here are those of the author and do not reflect the official policy or position of the U.S. Department of Defense or the U.S. federal government.

Copyright held by author.

Technology Strategy and Management

Competing in Emerging Markets

Considering the many different paths and unprecedented opportunities for companies exploring emerging markets.

THE BERLIN WALL fell in November 1989. This was a world-shaking event that triggered the disintegration of the Soviet Union, transforming our view of competition between nations. It also altered how we think about innovation-based competition in emerging markets. Now, 25 years later, we can take stock of these emerging markets.

The label “emerging markets” was coined by Antoine van Agtmael in the early 1980s. It remains alluring for investors, innovators, and governments. Yet it remains difficult to characterize them. Our view of emerging markets has changed over time. Precise understanding might help innovators and business managers compete more effectively. This column explores these claims.

Excitement about Emerging Markets

The interest in emerging markets is in part because they grow fast and have potential for further growth. Growth rates have slowed somewhat since Goldman Sachs started “dreaming with BRICs,” referring to the emerging economies of Brazil, Russia, India, and China.² But as developed economies see little growth in the age of austerity, emerging markets remain important destinations for global corporations to sell goods and services.



Fast growth comes because emerging markets provide low-cost locations for sourcing inputs, processing products, and delivering services. Firms led by giants such as Foxconn in electronics and Pou Chen Group in footwear generate wealth and jobs as suppliers to global brand owners (see my July 2011 *Communications* column).⁵ These firms are part of global value chains and have been catching up with firms from advanced economies. They have

accumulated capabilities in production but also in research. By 2010, U.S. *Fortune 500* companies had 98 R&D facilities in China and 63 in India. General Electric's health-care arm had multiple facilities.

Emerging markets also have innovative firms making new products and services. Sometimes these are dramatically cheaper than their Western equivalents: \$3,000 cars by Tata Motors, \$300 computers by Lenovo, and \$30 mobile



Association for
Computing Machinery

ACM Conference Proceedings Now Available via Print-on-Demand!

*Did you know that you can
now order many popular
ACM conference proceedings
via print-on-demand?*

Institutions, libraries and individuals can choose from more than 100 titles on a continually updated list through Amazon, Barnes & Noble, Baker & Taylor, Ingram and NACSCORP: CHI, KDD, Multimedia, SIGIR, SIGCOMM, SIGCSE, SIGMOD/PODS, and many more.

**For available titles and
ordering info, visit:
librarians.acm.org/pod**

handsets by HTC. These firms do not merely imitate U.S. and Japanese firms, but innovate to meet the needs of low-income consumers at the bottom of the pyramid and the rising middle classes. When prices are one-quarter to one-third of developed country prices, economies of scale and incremental process innovation is not enough. Profound understanding of local consumers' needs enables innovators to strip down unnecessary functionalities to reduce costs drastically. "Frugal innovation" focuses on product redesign and the invention of new models for production and distribution, not on technological breakthroughs.⁷ Healthcare is a hotbed for frugal innovation with General Electric competing with emerging market firms to develop low-cost portable medical equipment such as electrocardiogram and ultrasound machines.

"Local dynamos"¹ in emerging markets are suppliers and/or competitors of developed economy multinationals. Already in 2010, 17% of the *Fortune Global 500* companies (ranked by revenue size) were headquartered in emerging markets. McKinsey projects this will rise to 45% by 2025.³ Japanese companies with innovation in lean manufacturing posed threats to Western companies in the 1980s. Emerging market multinationals from China, India, Brazil, Mexico, Turkey, and many other countries pose similar threats, but with a material difference leading to greater uncertainty in the risk-reward equation.

Defining "Emerging Markets": Combining Economics and Politics

The *New York Times* called van Agtamel a marketing genius when reviewing his book, *The Emerging Market Century*. The label "emerging market" broke from seeing the Third World as underdeveloped or developing. Economic development had been about helping these countries with international aid and technology transfer. Developing countries protected domestic industries by erecting international trade and investment barriers. Some countries were firmly entrenched in the planned economy with the state owning the means of production. By contrast, emerging markets have free international trade and foreign direct

investment (FDI) and thriving private sectors as engines of growth.

The fall of the Berlin Wall in 1989 created "transition economies" in Central and Eastern Europe, moving from planned to capitalist models. There was a strong belief that capitalist economies would thrive only if entrenched in multi-party democracies with free elections. The Washington Consensus of the 1990s saw the International Monetary Fund (IMF) advance financing only if countries would move toward open, liberalized, and privatized economies. However, China, Russia, and Brazil endorsed options other than liberal democracy, giving rise to the label of "state capitalism"⁶ in which state-owned enterprises are an alternative to private-sector firms. Emerging market leaders believe that state capitalism is a sustainable alternative to liberal market capitalism. Economics and politics in emerging markets come in different colors and stripes.

Defining emerging markets today is more difficult than when the Berlin Wall fell. Low and middle-income countries with high growth potential remain part of the definition, but the expectation of 25 years ago that they are all moving toward liberal capitalist economies has changed. Some governments of emerging markets control resources and political freedom in ways liberal democracies do not.

Emerging Markets: Are They Behind or Ahead?

It is useful to examine the notion of innovation catch-up by emerging market companies. Some branches of science and medicine might give Western firms an edge, but a broader meaning of in-

**The interest in
emerging markets
is in part because
they grow fast
and have potential
for further growth.**

novation shows emerging market multinationals gaining ground in innovation that meets the needs of emerging market consumers. MTN from South Africa appears stronger than Western mobile operators in providing services to other African countries. Such south-south trade—the movement of goods and services from one emerging market to another—has been on the rise; it accounts for approximately half of total trade for China, and almost 60% of total trade for India and Brazil. Much is in agricultural products and minerals. But it is spreading to manufacturing, especially in renewable energy products (solar photovoltaic cells and modules, wind-powered generating sets, hydraulic turbines).

It is also useful to examine who is ahead or behind in implementing institutional reforms. Many emerging market countries suffer from unstable, weak, and inadequate institutions for tax collection, enforcement of intellectual property rights, and other functions. Yet global institutions are increasingly led by nationals from emerging market countries. These leaders set the global policy agenda and influence the global rules of the game. The fifth BRICS summit in March 2013 launched a New Development Bank to be headquartered in Shanghai as an alternative to international financial institutions such as the World Bank.

What This Means for Developed Economy Multinationals

To operate successfully in emerging markets, multinationals must navigate both host governments and the needs of local consumers. They must make clear choices about how best to do this. Archrivals Procter & Gamble and Unilever provide a good comparison. P&G is bigger (84 billion USD revenue) and more profitable (12.9% net profit margin) compared to Unilever (67 billion USD revenue and 9.6% net margin). Yet, Unilever has higher sales turnover (57% as compared to P&G's 38%) in fast-growing emerging markets.

P&G is global, operating in 80 countries and touching the lives of people in 180 countries. However, P&G centralized R&D and other functions to speed up product roll-out and to reduce the cost of operations (by 10 billion USD by 2016). This was done to

Global institutions are increasingly led by nationals from emerging market countries.

compete globally, but it means less adaptation to local markets and strategic decisions not to enter new markets. P&G focuses on 10 big emerging markets (Brazil, Russia, India, China, South Africa, Nigeria, Poland, Turkey, Mexico, and Indonesia).⁴

Unilever sells products in 190 countries, facilitated by a long history of adaptation and customization in local markets. It has built a large portfolio of relationships with local joint venture partners and distributors. This raises overall costs, but Unilever can reach market segments not normally touched by multinationals from high-income countries. Unilever packages shampoo and toothpaste in small sachets to be sold to low-income consumers, and its decentralized subsidiaries help develop a deep understanding of both consumers and host governments—the latter essential for anticipating regulatory and institutional changes.

Which will succeed best in emerging markets, P&G or Unilever? Any excellent company might combine aspects of these strategies, but the choices are difficult and involve trade-offs. The jury is still out on the best pathway to profitability and growth in emerging markets. P&G seeks to reduce cost by exploiting global economies of scale. Unilever seeks to adapt to local consumer needs through low-cost business innovation.

Conclusion

When the Berlin Wall fell 25 years ago, emerging markets were believed to become liberal market economies. Now we know better. Some emerging market countries follow different political arrangements, including state capitalism. Emerging markets present multinational corporations with

unprecedented opportunities, providing large markets for their products and low-cost locations for increasingly skilled labor.

Balanced against these opportunities are the challenges. There are two, both of which appear to give emerging market firms an upper hand over multinationals from developed economies. The first is the challenge of engaging in low-cost or frugal innovation that requires a profound understanding of unmet consumer needs. In China, Alibaba's lead over eBay, Baidu's lead over Google, and Xiaomi's lead over Apple are just a few examples to suggest this. One reason is that business innovation requires attention to distribution and logistics networks as well as to designing products and services.

The second challenge lies in understanding host governments. They impact business operations through granting of permits and licenses, protecting intellectual property, regulating foreign direct investment, and so on. Successful companies anticipate changes in regulation and rules. The unstable nature of the rule-making institutions in emerging markets adds uncertainty and risk. It pays for companies to understand these things.

Emerging markets require companies that understand frugal innovation and how to influence regulation and rules in localities that are unstable or unpredictable. Having antennae on the ground via subsidiaries in decentralized multinational corporations and/or through use of local emerging market firms as partners, seems to be a winning formula. ■

References

1. Chin, V. and Michael, D.C. BCG local dynamos: How companies in emerging markets are winning at home. Boston Consulting Group, 2014.
2. Goldman Sachs. Dreaming With BRICs: The path to 2050. *Global Economics Paper No. 99*, 2003.
3. MGI. Urban world: The shifting global business landscape. McKinsey Global Institute, 2013.
4. P&G chief hopes his mea culpa will help turn the tide. *Financial Times* (June 21, 2012).
5. Sako, M. Driving power in global supply chains. *Commun. ACM* 54, 7 (July 2011), 23–25.
6. The Rise of State Capitalism, special report on emerging market multinationals. *The Economist* (Jan. 21, 2012).
7. The World Turned Upside Down, special report on innovation in emerging markets. *The Economist* (Apr. 15, 2010).

Mari Sako (mari.sako@sbs.ox.ac.uk) is Professor of Management Studies at Saïd Business School, University of Oxford, U.K.

Copyright held by author.



DOI:10.1145/2736344

George V. Neville-Neil

 Article development led by **acmqueue**
queue.acm.org

Kode Vicious Raw Networking

*Relevance and repeatability.***Dear KV,**

The company I work for has decided to use a wireless network link to reduce latency, at least when the weather between the stations is good. It seems to me that for transmission over lossy wireless links we will want our own transport protocol that sits directly on top of whatever the radio provides, instead of wasting bits on IP and TCP or UDP headers, which, for a point-to-point network, are not really useful.

Raw Networking**Dear Raw,**

I completely agree that the best way to roll out a new networking service is to ignore 30 years of research in the area. Good luck.

Second only to operating system developers—all of whom want to rewrite the scheduler (see “Bugs and Bragging Rights,” second letter, at <http://bit.ly/1yGXHV9>)—are the networking engineers and developers who want to write their own protocol. “If only we could go at it with a clean sheet of paper, we could do so much better than the ARPANET, since that was designed for old, crappy hardware, and ours is shiny and new.” That statement is both true and false, and you had better be damned sure about which side of the Boolean logic your idea lies before you write a single line of new code.

The Internet protocols are not the be all and end all of networking, but they have had more research and test-



ing time applied to them than any other network protocols currently in existence. You say you are building a wireless network with—I am sure—the highest-quality gear you can buy. Wireless networks are notoriously lossy, at least in comparison to wired networks. And it turns out there has been a lot of research done on TCP in lossy environments. So although you will pay an extra 40 bytes per packet to transport data over TCP, you might get

some benefit from the work done—to tune the bandwidth and round-trip time estimators—that will exist in the nodes sending and receiving the data.

Your network is point-to-point, which means you do not think you care about routing. But unless all the work is always going to be carried out at one or the other end of this link, you are eventually going to have to worry about addressing and routing. It turns out that someone thought

about those problems, and they implemented their ideas in, yes, the Internet protocols.

The TCP/IP protocols are not just a set of standard headers, they are an entire system of addressing, routing, congestion control, and error detection that has been built upon for 30 years and improved so users can access the network from the poorest and most remote corners of the network, where bandwidth is still measured in kilobits and latencies exceed half a second. Unless you are building a system that will never grow and never be connected to anything else, you had better consider whether or not you need the features of TCP/IP.

I am all for clean-sheet research into networking protocols, there are many things that have not been tried and some that have been, but did not work at the time. Your letter implied not so much research, but rollout, and unless you have done your homework, this type of rollout will flatten you and your project.

KV

Dear KV,

You write about the importance of testing, but I have not seen anything in your columns on how to test. It is fine to tell everyone testing is good, but some specifics would be helpful.

How Not Why

Dear How,

The weasel's way out of this response would be to say there are too many ways to test software to give an answer in a column. After all, many books have been written about software testing. Most of those books are dreadful, and for the most part, also theoretical. Anyone who disagrees can send me email with their favorite book on software testing and I will consider publishing the list or trashing the recommendation. What I will do here is describe how I have set up various test labs for my specific type of testing, and maybe this will be of some use.

There are two requirements for any testing regimen: *relevance* and *repeatability*. Test-driven development is a fine idea, but writing tests for the sake of writing tests is the same as measur-

When I see a poor testing setup I should be prepared to see poor code as well.

ing a software engineer's productivity in KLOC. To write tests that matter, test developers have to be familiar enough with the software domain to come up with tests that will both confirm that the software works and which also attempt to break the software. Much has been written about this topic, so I am going to switch gears to talk about repeatability.

Tests are considered repeatable when the execution of two different tests on the same system do not interfere with one another. A concrete example from my own work is the population of various software caches—such as routing and ARP tables—that might speed up the second test in a series of tests of packet forwarding. To achieve repeatability, the system or person running the test must have complete control over the environment in which the test runs. If the system being tested is completely encapsulated by a single program with no side effects, then running the program repeatedly on the same inputs is a sufficient level of control. But most systems are not so simple.

Working from the concrete example of testing a firewall: To test any piece of networking equipment that passes packets from one network to another, you need at least three systems, a source, sink, and the device under test (DUT, in test parlance.) As I pointed out earlier, repeatability of tests requires a level of control over the systems being tested. In our network testing scenario, that means each system requires at least two interfaces and the DUT requires three. The source and sink need both a control interface and the interface on which packets will be either sent to or received from the DUT. "Why can't

we just use the control interfaces to source and sink the packets?" I hear you cry, "Wiring all that stuff is complicated and we have three computers on the same switch, we can just test this now." The way it works is the control and test interfaces must be distinct on all the systems to prevent interference during the test. No matter what you are testing, you must ensure you reduce the amount of outside interference unless that is what you are intending to test. If you want to know how a system reacts with interference, then set up the test to introduce the interference, but do not let interference show up out of nowhere. In our specific networking case, we want to retain control over all three nodes, no matter what happens when we blast packets across the firewall. Retaining control of a system under stress is non-trivial.

Another way to maintain control over the systems is to have access to a serial or video console. This requires even more specialized wiring than just a bunch more network ports, but it is well worth it. Often, bad things happen, and the only way to regain control over the systems is via a console login.

The ultimate fallback for control is the ability to remotely power-cycle the system being tested. Modern servers have an out-of-band management system, such as IPMI, that allows someone with a user name and password to remotely power-cycle a machine as well as do other low-level system management tasks including connecting to the console. Whenever someone wants me to test networked systems in the way I am describing, I require them to have either out-of-band power management via a network-connected power controller or IPMI on the systems in question. There is nothing more frustrating during testing than having a system wedge itself and having to either walk down to the data center to reset it. Or worse, having your remote hands have to do it for you. The amount of time I have wasted in testing because someone was too cheap to get IPMI on their servers or put in a proper power controller could have been far better spent killing the brain cells that had absorbed the same company's poorly

written code. It seems that inattention to detail is pervasive, and when I see a poor testing setup I should be prepared to see poor code as well.

At this point, we now know that we have to retain control over the systems—and we have several ways to do that via separate control interfaces—and ultimately, we have to have control over the system's power. The next place that most test labs fall down is in access to necessary files.

Once upon a time a workstation company figured out they could sell lots of cheap workstations if they could concentrate file storage on a single, larger, and admittedly more expensive, server. Thus was born the Network File System, the much maligned, but still relevant, way of sharing files among a set of systems. If your tests can in any way destroy a system, or if upgrading a system with new software removes old files, then you need to be using some form of networked file system. Of late I have seen people try to handle this problem with distributed version control systems such as Git, where the test

code and configurations are checked out onto the systems in the test group. That might work if everyone were diligent about checking in and pushing changes from the test system. But in my experience, people are never that diligent, and inevitably someone upgrades a system that had crucial test results or configuration changes on it. Use a networked file system and it will save whatever hair you have left on your head. (I should have learned this lesson sooner.) Ensure the networked file system traffic goes across the control interfaces and not the test interfaces. That should go without saying, but in test lab construction, much of what I think could go without saying needs to be said.

At this point we have fulfilled the most basic requirements of a networking test system: We have control over all the systems, and we have a way to ensure all the systems can see the same configuration data without undue risk of data loss. From here it is time to write the automation that controls these systems. For most testing scenarios, I tend to just reboot all

the systems on every test run, which clears all caches. That is not the right answer for all testing, but it definitely reduces interference from previous runs.

KV

Related articles on queue.acm.org

Orchestrating an Automated Test Lab

Michael Donat

<http://queue.acm.org/detail.cfm?id=1046946>

The Deliberate Revolution

Mike Burner

<http://queue.acm.org/detail.cfm?id=637960>

Kode Vicious Unleashed

George Neville-Neil

<http://queue.acm.org/detail.cfm?id=1046939>

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

Copyright held by author.



REQUIREMENTS

- A Bachelor degree or the equivalent in Computer Science or related majors
 - Excellent academic credentials demonstrated by GPA scores
 - Competitive GRE test scores
- Prior publications and an MS degree in CS (or a related area) are a plus.

HOW TO APPLY:

All applications must be submitted online at <http://www.students.graduate.ucf.edu/gradonlineapp/>

GRADUATE STUDY AT UCF IN ORLANDO

Computer Science at the University of Central Florida (UCF) is looking for graduate students who are motivated to conduct cutting-edge research and pursue excellence. Successful applicants will be offered 5 years of support (assuming satisfactory progress) in the form of either a Graduate Teaching Assistantship or Graduate Research Assistantship. These positions will be providing financial support, including tuition waivers, and competitive stipends. US citizens are highly encouraged to apply.

UCF is located in Orlando FL – “The City Beautiful.” Orlando has hosted several major academic conferences in recent years, including ACM Multimedia. UCF is the largest university in the United States by undergraduate enrollment and the country's second-largest by total enrollment. Both Orlando and UCF have grown from supporting the space program to supporting a thriving high-technology industry, including many software startups.

Computer Science at UCF is performing ground-breaking research in a wide range of research areas, including computer vision, image and video processing, machine learning, AI, virtual reality, HCI, computer graphics, software engineering and systems, database, parallel computation, networking and mobile computing, computer security, bioinformatics and systems biology, theory, algorithms, and quantum computing. See <http://www.cs.ucf.edu>.

These research areas have been sponsored by federal government agencies and local governments, as well as established close collaborations with the leading companies such as Google, Sarnoff, Microsoft, IBM, GE, and Siemens. The Computer Science program has internationally-recognized faculty, including 5 Fellows of the IEEE.

We are also very proud of many of our students' achievements, for example, National Collegiate Cyber Defense Championship, a prestigious national award for cybersecurity in 2014, and successfully competing at the World Finals of the Association of Computing Machinery's International Collegiate Programming Contest.

Interview

An Interview with Juris Hartmanis

A pioneer in the field of computational complexity theory reflects on his career.

ACM FELLOW JURIS HARTMANIS, recipient of the 1993 A.M. Turing Award with Richard E. Stearns, has made fundamental contributions to theoretical computer science—particularly in the area of computational complexity—for over 50 years. After earning a Ph.D. in Mathematics from Caltech, he developed the foundations of this new field first at General Electric Research Laboratory and then at Cornell University. He says “Computational complexity, the study of the quantitative laws that govern computation, is an essential part of the science base needed to guide, harness, and exploit the explosively growing computer technology.”

Noted historian and *Communications Viewpoints* board member William Aspray conducted an extended oral interview of Hartmanis in his Cornell University office in July 2009. The complete transcript of this interview is available in the ACM Digital Library; presented here is a condensed and highly edited version designed to whet your appetite.

—Len Shustek

An Unusual Early Life

I was born on July 5, 1928 in Riga, the capital of Latvia, and was very fortunate to have been born into a prominent Latvian family. My father was the Chief of Staff of the Latvian army, and my early childhood was secure, pleasant,



Juris Hartmanis.

and interesting. I attended the excellent French Lycee there, expecting to follow my father into a military career. I am surprised how much motivation and insight they gave besides teaching the basic subject matters.

That good life unfortunately changed when I was about 12 years old. The Soviets occupied Latvia, and in the winter of 1940 my father was arrested. We did not know for years what happened to him. Only after the Soviet

Union collapsed in the 1990s and their archives were opened did we find out that my father has been taken to Moscow, tried, convicted, and executed. The Soviet occupation really was very, very brutal.

When Riga fell to the Soviets in 1944, we left by ship and moved to the university town of Marburg an der Lahn in Germany, and I enrolled in a German high school. But this was 1944 and there wasn't much of a school year;

ACM Transactions on Accessible Computing



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

www.acm.org/taccess
www.acm.org/subscribe



Association for
Computing Machinery

every time allied bombers approached there were air raid alerts, and everybody had to proceed to air raid shelters. By 1945 everybody knew the war was lost, and German soldiers surrendered en masse in a disciplined manner.

When the war was over I attended Latvian high schools, first in the English occupation zone, then in the American zone. Travel was very difficult. There were bombed-out bridges where trains had passed over valleys; trains now stopped on one side and you carried your luggage down the valley and up the other side where another train was supposed to be waiting.

Germany at that time was really a sad place and life in general was difficult. But the schoolteachers were superb. They were highly educated and motivated, and often were professors from the Latvian university. I was inspired to continue my education. I studied physics for two-and-a-half years at the University in Marburg, and enjoyed it immensely.

These were hard times, sad times. Just about every institute was stripped of much of its equipment during the wartime. But there is a certain advantage to have to improvise and overcome difficulties. I did not possess any physics textbooks—they were very hard to find, and anyway I would not have had the money to buy them—so I compensated by taking very detailed notes in lectures. I got good grades and strong recommendations from the professors.

Going to the United States

Though I felt quite comfortable with the language and enjoyed my studies, Germany wasn't our country. Not only that, the outlook for a successful career and life in Germany did not look very promising. Among Latvians in Germany, the top preference was to emigrate to America or Canada. My mother had friends from Latvia who had arrived in the states earlier and lived in Independence, Missouri, so that's where we went.

After our arrival in Independence I worked for Gleaner Harvester Company, which built agricultural combines. When production was cut back and the new hires were let go, I went to work for Sheffield Steel in Kansas City as a steel worker. I even became a union member. It was interesting.

During the summer I went to the University of Kansas City. They gave me a credit of 128 units for my five semesters in Marburg, and decided that I had the equivalent of a bachelor's degree. They also gave me a scholarship and admitted me as a graduate student, which surprised me since I had studied in Marburg for only two-and-a-half years.

I was delighted to be accepted as a graduate student, but there was a problem: there was no graduate program in physics. There was a graduate program in mathematics, so they said, "You will be completely happy studying mathematics." And I was.

My mother and I lived with and worked for a very famous physician. I was a butler and served dinner every day. I washed his two cars: the 12-cylinder Lincoln and the 8-cylinder Cadillac. I drove his wife's old mother to church every Sunday. Things like that.

I was a straight-A student. The University of Kansas City at that time was in no sense an elite university, but it had a respectable mathematics department and it was a good place to study and improve my English.

After I received my master's degree at the University of Kansas City I applied to several universities, among them CalTech.

Why Caltech?

I was told that Caltech really was an elite school, one of the best schools in physics and mathematics. Not only that, Caltech gave me \$90 to get there, and offered a teaching assistant's job. They read my situation very well. So my mother and I left Kansas City and drove my first car across the country to Caltech.

Caltech admitted me not really knowing what I wanted to do, after having first done physics in Germany and then mathematics in Kansas City. They said "You look like an applied mathematician. We do not have an applied mathematics track, but we believe you will be perfectly happy doing mathematics." Since I'd never had a course in applied mathematics I said "Okay," and my fate was decided.

Caltech was first-class. It just bubbled with creativity, with scientific work. Nobel Prize winners were running all around: Linus Pauling, Carl Anderson of cosmic rays, Millikan of the

famous measurement of the electron charge with the oil drop experiment. Richard Feynman, who received the Nobel Prize in 1965, was one of the most popular of the teachers and scientists; physics graduate students just adored him. Sometime during my stay there, James Watson of DNA fame showed up. There was such intense research activity that you almost felt “I haven’t done anything today. I should. Everybody else is producing great results.”

I enjoyed my stay at CalTech very much. California was a great place, and life in Pasadena was exciting. There was Hollywood, the beaches, the mountains, the Rose Parade—life was just beautiful. I enjoyed it even more when I met Elly Rehwald, who is now my wife; just recently we celebrated our 50th wedding anniversary.

Collaborating with Robert Dilworth

I do not recall exactly how I selected or was selected to work with Professor Robert Dilworth, an algebraist who worked in lattice theory. After I passed my qualifying exams Dilworth suggested I work on the partition lattice embedding problem: showing that partition lattices contain any other lattice as a sublattice.

Well, I worked on it. It was interesting, but it was hard. I realize now that the partition lattice embedding theorem was a first-class problem that after decades had not been solved.

I struggled with it, and finally said, “Let’s think around it. Let’s look at the terrain around the problem.” I invented generalized partitions, and realized they behaved like lines in geometry. There is at least some intuition about lines and subspaces. I started developing techniques to solve the embedding problem for generalized partition lattices.

I did not keep a log, so I do not know exactly when I started working in this problem and when I solved it. My recollection is that it was a very intensive period of research. I gained real insights about the lattice of geometries and solved the embedding problem. I gave a seminar on my results that was very well received. Dilworth urged me to write up my results, and I was told that I had my thesis.

This research experience taught me a tremendous amount. First, how to do research; that thinking

“Caltech admitted me not really knowing what I wanted to do.”

about the whole intellectual terrain around problems is important. It is like climbing mountains: there may be different ways to the top than the steepest slope.

I also discovered, during my education and early research experience, that you do not have to know that much to do research. With a good background education, one can very quickly absorb the necessary specialized knowledge for a specific research area. I think what Caltech really taught me was to be independent, and to try to go to the essence of a problem.

Moving East, to Cornell

During my last year at CalTech, professor Bob Walker from Cornell visited. After discussions with Dilworth and on his recommendation, my friend Johnny Johnson and I were, on the spot, given verbal offers to come to Cornell as instructors of mathematics. In those days, you did not start as an assistant professor; you started as an instructor. Without having seen Cornell, both of us accepted the offers.

The Cornell campus is charming. We were immensely taken by the beauty of the surrounding area and the campus itself. In the math department we were very junior members and started by teaching calculus. It was a very friendly environment. We easily made friends and really felt like apprentice professors.

Next: General Electric

Dick Shuey, the General Electric Research Lab manager in nearby Schenectady, was a very far-sighted person who had convinced GE that they should become a great computer company, and that a science base is needed for that. He traveled around finding people to do research in “information,” which we quickly translated as

computer science. He was immensely impressed by Shannon’s work, and so was I. Shuey visited me in the math department and convinced me to take a summer job.

The research traditions there are old, and the laboratory has a string of successes. The information section was still quite small, and so any newcomer was very welcome. By the end of the summer I published a paper on linear coding networks in the IRE (later IEEE) *Transactions on Circuits*. The experience was very encouraging—that in such a short time I managed to produce a refereed paper that was accepted—and it opened up some new areas for me to think about.

That period was almost ideal. The computer science research people were quite free; I never had any explicit orders to work on particular problems.

Under Shuey’s guidance, one of the very early papers I studied was Shannon and Weaver’s book on information theory, *Mathematical Theory of Communication*. It impressed me immensely that from what seemed to me vague concepts such as channel capacity and coding for error-correcting codes, one could make a mathematical theory that guides you in how to communicate and gives you quantitative measures of what is the best you can do with a certain channel and with a certain information source. Unfortunately, in my attempts to apply it to computing there was no success. I played with information theoretic concepts, and I wrote a paper about entropy, but that was exploratory work that did not lead to very much.

Doing Corporate Research

Our job was just research, full-time, with no teaching obligations. We read whatever we could lay our hands on that said something about computing.

At Caltech I had not been exposed to concepts of computability, undecidability, or Turing machines. At the lab we read all those classic papers, and I realized how absolutely beautiful those ideas are. I had large photographs of Shannon, Turing, and von Neumann in my office; those were the people I admired and thought to be the pioneers in computer science.

Dick Stearns and I started working together during the summer and hit

it off really well. When he finished his Ph.D. and joined the research lab as a research mathematician, we worked day in and day out together, sitting and staring at each other in his office or mine, shouting and hollering about the other's ignorance for not understanding the subtlest points of computability. We did a lot of work on finite automata, particularly in decomposing finite automata into smaller ones from which they could be realized. In 1966 we published a book summarizing our work, *Algebraic Structure Theory of Sequential Machines*.

When one looks at the early years of theoretical work in computer science, there was a lot of switching theory, which really was how to design circuits. We worked on code assignment, optimal naming of finite automata states, and related problems. But soon the research moved away from finite devices and started exploring formal languages, push-down automata, Turing machines, design and analysis of algorithms, computational complexity, and so forth. This was a turn to more realistic models and problems about real computation.

Starting Computational Complexity

In the very early 1960s, when we had really well understood Turing's work, we read a paper by Yamada on real-time computation. Yamada was interested in computing when the Turing machine had to print the digits at a steady pace and could not slow down. He showed quite easily that there were recursive sequences that could not be so computed.

Yamada studied just this one class with a time bound. But we thought there should be a theory about all complexity classes, and that every computation should fall in some class. We started doing generalized computational complexity. Among the key ideas was the realization that every solution to a class of problems should have a computation bound as the problem grows in size. We simply defined computational complexity classes by a function, like say n^3 , that encompassed all problems that can be solved in n^3 steps—which we referred to as time—for problem size n . We proved a general hierarchy theorem

that showed that for nice bounds there are computations that can be done exactly in the given bound and not within a smaller bound. We also showed that for any given computable bound there were problems not computable in the given bound. Today this approach is known as asymptotic complexity.

In April 1963 we finished the paper that later earned us the Turing Award, on the computational complexity of algorithms. We originally submitted it to the *Journal of the ACM*, but there was lots of mathematics and we worried that not too many people would care to study it. So we published it in the *Transactions of the American Mathematical Society* instead. The first public presentation was in 1964 at the IEEE Annual Symposium on Switching Theory and Logical Design, and it was very well received. We were confident that we had opened a new and important research area, and we were totally committed to exploring computational complexity.

Expanding the Field

We started exploring other complexity measures besides time. We studied, along with Phil Lewis, tape and memory bounded computational complexity classes, now also known as space-bounded complexity classes. We proposed a new Turing machine model that had a read-only two-way input tape and a separate working tape for the computation.

Many computer scientists joined this research area. For example, our context-free language recognition $(\log n)^2$ algorithm led Savage to generalize it and prove a beautiful relation between deterministic and

“When I joined, GE had all the things in place to become a dominant computer designer and manufacturer.”

nondeterministic tape bounded computations. The work in complexity theory started spreading.

As the group of people working in this area grew, we felt a need for a conference and publication dedicated to computational complexity. In 1986 we organized the first conference, “Structure in Complexity Theory”—“structure” because we were interested in how all these complexity classes relate to each other.

It is quite amazing that in spite of the 45 years of impressive progress understanding the complexity of computations, we still cannot prove that P is different from NP and NP is different from PSPACE, the problems that can be solved with a polynomial-length tape. Intuitively, all these classes look to be different, but no proof has been found. Until we settle the separation problem for these classes, particularly Cook's notorious P versus NP problem, we have not fully understood complexity of computations. But I do believe that they are provable, and will be solved.

What Happened at GE

When I joined, GE had all the things in place to become a dominant computer designer and manufacturer. We hoped that our work would eventually be the foundation for, or help with, the computer business. But GE failed to exploit these early successes and their computer effort was fumbled away. GE had the philosophy that a good manager can manage anything, and that was proved absolutely wrong in the computer field. It was a great failure and a great disillusionment for my colleagues and me.

I had really enjoyed Cornell before, so in 1965 I accepted their offer to become a full professor with tenure, and to be chair of their new graduate department in computer science. There were a number of other computer science departments that were being formed or planned, and the excitement in computer science almost perceptibly had shifted toward education.

Creating the Cornell Department of Computer Science

Our greatest problem was finding faculty. The most important part is to get the best possible people you can. Quality, quality, quality.

We decided to start with a very light teaching load: one course per semester plus participation in a seminar. Not only that, our Sloan Foundation support allowed us to pay better salaries than were paid for the same ranks in the math department and some other computer science departments. We created a very strong department, although we were really overloaded in theory.

We worked very hard on the intellectual environment, the cohesiveness of the department. I decided not to have regular faculty meetings. We all met at the Statler Club, and after a quick lunch, we gathered in a corner niche of the Cornell Faculty Club for coffee and discussions. All departmental issues were discussed in these meetings. I, as chair, listened and argued and listened some more. If I felt a consensus emerging, I said so, and then I implemented the decisions. From the very beginning we had the tradition that everybody is to be heard.

Other Jobs: Pondering the Future of Computer Science

What is computer science and engineering? What should the field be doing? What does the field need in order to prosper? That was the mandate for “Computing the Future,” a 1992 study by the Computer Science and Telecommunications Board of the National Academies.

We did argue. It is amazing when you put a bunch of bright, successful, good scientists in one room and ask them, “Tell me what computer science is.” A very lovely, lively discussion.

We made recommendations: sustain the core effort in computer science, improve undergraduate education, and broaden the scope of computer science. This meant broadening interdisciplinary work as well as expanding in new applications; we wanted to see every computer science Ph.D. program require a minor in some other discipline.

I think it is a good report, but there was some controversy. John McCarthy from Stanford University got very upset; he felt that AI did not have the prominent role it should have. The controversy died down, but I think it did bring some more publicity to the report. A number of people have said they were delighted that it fought for a

broader agenda. At the end the report was well received by the CS community.

A Tour of Duty at the National Science Foundation

In 1996 I went to NSF as Assistant Director of the Directorate for Computer and Information Science and Engineering, CISE. Absolutely loved Washington, and loved the job. It was a kind of twofold job. One, run CISE: worry about the distribution of funds, and what areas to support. I felt strongly that the program managers just did not exercise enough of their power in making their own decisions about what will or will not get funded. They really have a fair amount of power to do that. Complaints can be leveled that the program managers do not do enough risky research support and enough interdisciplinary funding. But, for example, I closed down an engineering program that was supporting university chip design, because by that time there were other venues.

The other job is to represent computer science and represent CISE within NSF, where you compete with your fellow assistant directors for funding. You spend a lot of time telling, in short and long paragraphs, about computer science. You basically are a spokesman for computer science.

At CISE I did a thing that had not been done before ever: I reviewed every bloody program. I sat down individually with every program manager and they explained what the program was doing, what research it was supporting. I told all my program and division managers, “I need nuggets. I need crisply explainable research results.”

CISE ratings were lower than a number of other directorates on the average. I think computer scientists probably still do not have as uniform an understanding of each other as the physicists do, for example. Physicists will argue about different projects, but I think they have a firmer way of judging each other’s research proposals. Computer science is new. That’s no real excuse, but it’s growing fast, it’s changing.

I stayed a little bit over two years. Within NSF it’s a very delicate relationship, how hard you try to convince the director that you should get the biggest slice. CISE did quite well, but when CISE started it was very small and its

budget had to be built up to meet the expanding CS needs. That was not an easy process, to argue for the recognition of computer science in a world dominated by physicists.

We were concerned, and we still should be concerned, that we are not attracting enough women. We are losing out on talent. We were puzzled, and we put extra money, while I was there, in fellowships for women. I am still surprised that there are more women in mathematics, percentage wise, than in computer science.

Summing Up

I was once asked, “Which of your two research achievements (not my words, his words) do you think is more important: the structure of finite automata, or complexity theory?” Without a second’s thought I said, “Complexity.”

Finite automata were fun. With Dick Stearns, some parts were just like playing a big, interesting game. There were some very unintuitive results. A number of people joined in that kind of fun, but I think that is more of parlor-type mathematics.

But almost all computer science problems involve algorithms—and therefore computational complexity problems. For example, computational complexity theory has put cryptography on a different base. I think every computer scientist should have some intuitive grasp of computational complexity.

Recently there have been very interesting results about the complexity of approximate results to NP and other problems that we believe not to be feasible computable. Many of these problems have easily computable approximations, but many others do not. In design of algorithms it has helped tremendously in knowing what can and cannot be done. In some cases we do not have that insight yet, but in other cases it helped, and better algorithms have emerged.

These are deep results and reveal very interesting connections between different aspects of computation, although there are important unsolved problems that have been around for quite a long time. 

Len Shustek (shustek@computerhistory.org) is the chairman of the Computer History Museum.

Copyright held by author.

Viewpoint

Who Builds a House without Drawing Blueprints?

Finding a better solution by thinking about the problem and its solution, rather than just thinking about the code.

I BEGAN WRITING programs in 1957. For the past four decades I have been a computer science researcher, doing only a small amount of programming. I am the creator of the TLA+ specification language. What I have to say is based on my experience programming and helping engineers write specifications. None of it is new; but sensible old ideas need to be repeated or silly new ones will get all the attention. I do not write safety-critical programs, and I expect that those who do will learn little from this.

Architects draw detailed plans before a brick is laid or a nail is hammered. But few programmers write even a rough sketch of what their programs will do before they start coding. We can learn from architects.

A blueprint for a program is called a *specification*. An architect's blueprint is a useful metaphor for a software specification. For example, it reveals the fallacy in the argument that specifications are useless because you cannot generate code from them. Architects find blueprints to be useful even though buildings cannot be automatically generated from them. However, metaphors can be misleading, and I do not claim that we should write specifications just because architects draw blueprints.

The need for specifications follows from two observations. The first is that



it is a good idea to think about what we are going to do before doing it, and as the cartoonist Guindon wrote: "Writing is nature's way of letting you know how sloppy your thinking is."

We think in order to understand what we are doing. If we understand

something, we can explain it clearly in writing. If we have not explained it in writing, then we do not know if we really understand it.

The second observation is that to write a good program, we need to think above the code level. Programmers

spend a lot of time thinking about how to code, and many coding methods have been proposed: test-driven development, agile programming, and so on. But if the only sorting algorithm a programmer knows is bubble sort, no such method will produce code that sorts in $O(n \log n)$ time. Nor will it turn an overly complex conception of how a program should work into simple, easy to maintain code. We need to understand our programming task at a higher level before we start writing code.

Specification is often taken to mean something written in a formal language with a precise syntax and (hopefully) a precise semantics. But formal specification is just one end of a spectrum. An architect would not draw the same kind of blueprint for a toolshed as for a bridge. I would estimate that 95% of the code programmers write is trivial enough to be adequately specified by a couple of prose sentences. On the other hand, a distributed system can be as complex as a bridge. It can require many specifications, some of them formal; a bridge is not built from a single blueprint. Multithreaded and distributed programs are difficult to get right, and formal specification is needed to avoid synchronization errors in them. (See the article by Newcombe et al. on page 66 in this issue.)

The main reason for writing a formal spec is to apply tools to check it. Tools cannot find design errors in informal specifications. Even if you do not need to write formal specs, you should learn how. When you do need to write one, you will not have time to learn how. In the past dozen years, I have written formal specs of my code about a half dozen times. For example, I once had to write code that computed the connected components of a graph. I found a standard algorithm, but it required some small modifications for my use. The changes seemed simple enough, but I decided to specify and check the modified algorithm with TLA+. It took me a full day to get the algorithm right. It was much easier to find and fix the errors in a higher-level language like TLA+ than it would have been by applying ordinary program-debugging tools to the Java implementation. I am not even sure I would have found all the errors with those tools.

Writing formal specs also teaches

We need to understand our programming task at a higher level before we start writing code.

you to write better informal ones, which helps you think better. The ability to use tools to find design errors is what usually leads engineers to start writing formal specifications. It is only afterward that they realize it helps them to think better, which makes their designs better.

There are two things I specify about programs: what they do and how they do it. Often, the hard part of writing a piece of code is figuring out what it should do. Once we understand that, coding is easy. Sometimes, the task to be performed requires a nontrivial algorithm. We should design the algorithm and ensure it is correct before coding it. A specification of the algorithm describes how the code works.

Not all programs are worth specifying. There are programs written to learn something—perhaps about an interface that does not have an adequate specification—and are then thrown away. We should specify a program only if we care whether it works right.

Writing, like thinking, is difficult; and writing specifications is no exception. A specification is an abstraction. It should describe the important aspects and omit the unimportant ones. Abstraction is an art that is learned only through practice. Even with years of experience, I cannot help an engineer write a spec until I understand her problem. The only general rule I have is that a specification of what a piece of code does should describe everything one needs to know to use the code. It should never be necessary to read the code to find out what it does.

There is also no general rule for what constitutes a “piece of code” that requires a specification. For the pro-

Calendar of Events

April 13–16

CPS Week '15: Cyber Physical Systems Week 2015,
Co-Sponsored: Other Societies,
Seattle, WA,
Contact: Jie Liu,
Contact Email: liuj@microsoft.com

April 13–17

Symposium on Applied Computing, Salamanca, Spain,
Sponsored: ACM/SIG,
Contact: Roger Wainwright,
Contact Email: rogerw@utulsa.edu

April 14–16

HSCC'15: 18th International Conference on Hybrid Systems: Computation and Control (part of CPS Week),
Seattle, WA,
Sponsored: ACM/SIG,
Contact: Sriram Sankaranarayanan,
Contact Email: srirams@gmail.com

April 14–16

ICCPs '15: ACM/IEEE 6th International Conference on Cyber-Physical Systems (with CPS Week 2015),
Seattle, WA,
Co-Sponsored: Other Societies,
Contact: Ian Mitchell,
Contact Email: mitchell@cs.ubc.ca

April 14–16

IPSN '15: The 14th International Conference on Information Processing in Sensor Networks (co-located with CPS Week 2015),
Seattle, WA,
Co-Sponsored: Other Societies,
Contact: Bhaskar Krishnamachari,
Contact Email: bkrishna@usc.edu

April 18–23

CHI '15: CHI Conference on Human Factors in Computing Systems,
Seoul, Republic of Korea,
Sponsored: ACM/SIG,
Contact: Jinwoo Kim,
Contact Email: create2gether@gmail.com

April 21–24

EuroSys '15: Tenth EuroSys Conference 2015,
Bordeaux, France,
Sponsored: ACM/SIG,
Contact: Laurent Reveillere,
Contact Email: reveillere@labri.fr

Optimization as Motion Selection Principle in Robot Action

Privacy Behaviors After Snowden: The Brief Impact of Exposed State Surveillance

Parallel Processing with Promises

There Is No Now

Decoding Femininity in Computer Science in India

Managing Your Digital Life

Can Traditional Programming Bridge the Ninja Performance Gap for Parallel Computing Applications?

Teach Foundational Language Principles

Plus the latest news about trading accuracy for energy, robots with the human touch, and data/science journalism.

programming I do, it may be a collection of fields and methods in a Java class, or a tricky section of code within a method. For an engineer designing a distributed system, a single spec may describe a protocol that is implemented by code in multiple programs executed on separate computers.

Specification should be taught in school. Some universities offer courses on specification, but I believe that most of them are about formal specification languages. Anything they teach about the art of writing real specs is an accidental by-product. Teachers of specification should write specifications of their own code, as should teachers of programming.

Computer scientists believe in the magical properties of language, and a discussion of specification soon turns to the topic of specification languages. There is a standard language, developed over a couple of millennia, for describing things precisely: mathematics. The best language for writing informal specifications is the language of ordinary math, which consists of precise prose combined with mathematical notation. (Sometimes additional notation from programming languages can be useful in specifying how a program works.) The math needed for most specifications is quite simple: predicate logic and elementary set theory. This math should be as natural to a programmer as numbers are to an accountant. Unfortunately, the U.S. educational system has succeeded in making even this simple math frightening to most programmers.

Math was not developed to be checked by tools, and most mathematicians have little understanding of how to express things formally. Designers of specification languages usually turn to programming languages for inspiration. But architects do not make their blueprints out of bricks and boards, and specifications should not be written in program code. Most of what we have learned about programming languages does not apply to writing specifications. For example, information hiding is important in a programming language. But a specification should not contain lower-level details that need to be hidden; if it does, there is

something wrong with the language in which it is written. I believe the closer a specification language comes to ordinary mathematics, the more it aids our thinking. A language may have to give up some of the elegance and power of math to provide effective tools for checking specs, but we should have no illusion that it is improving on ordinary mathematics.

Programmers who advocate writing tests before writing code often believe those tests can serve as a specification. Writing tests does force us to think, and anything that gets us to think before coding is helpful. However, writing tests in code does not get us thinking above the code level. We can write a specification as a list of high-level descriptions of tests the program should pass—essentially a list of properties the program should satisfy. But that is usually not a good way to write a specification, because it is very difficult to deduce from it what the program should or should not do in every situation.

Testing a program can be an effective way to catch coding errors. It is not a good way to find design errors or errors in the algorithm implemented by the program. Such errors are best caught by thinking at a higher level of abstraction. Catching them by testing is a matter of luck. Tests are unlikely to catch errors that occur only occasionally—which is typical of design errors in concurrent systems. Such errors can be caught only by proof, which is usually too difficult, or by exhaustive testing. Exhaustive testing—for example, by model checking—is usually possible only for small instances of an abstract specification of a system. However, it is surprisingly effective at catching errors—even with small models.

The blueprint metaphor can lead

If we do not start with a specification, every line of code we write is a patch.

us astray. Blueprints are pictures, but that does not mean we should specify with pictures. Anything that helps us think is useful, and pictures can help us think. However, drawing pictures can hide sloppy thinking. (An example is the classic plane-geometry “proof” that all triangles are isosceles.) Pictures usually hide complexity rather than handling it by abstraction. They can be good for simple specifications, but they are not good for dealing with complexity. That is why flowcharts were largely abandoned decades ago as a way to describe programs.

Another difference between blueprints and specifications is that blueprints get lost. There is no easy way to ensure a blueprint stays with a building, but a specification can and should be embedded as a comment within the code it is specifying. If a tool requires a formal specification to be in a separate file, a copy of that file should appear as a comment in the code.

In real life, programs often have to be modified after they have been specified—either to add new features, or because of a problem discovered during coding. There is seldom time to rewrite the spec from scratch; instead the specification is updated and the code is patched. It is often argued that this makes specifications useless. That argument is flawed for two reasons. First, modifying undocumented code is a nightmare. The specs I write provide invaluable documentation that helps me modify code I have written. Second, each patch makes the program and its spec a little more complicated and thus more difficult to understand and to maintain. Eventually, there may be no choice but to rewrite the program from scratch. If we do not start with a specification, every line of code we write is a patch. We are then building needless complexity into the program from the beginning. As Dwight D. Eisenhower observed: “No battle was ever won according to plan, but no battle was ever won without one.”

Another argument against specification is that the requirements for a program may be too vague or ill-defined to be specified precisely. Ill-defined requirements mean not that we do not have to think, but that we have to think even harder about what a program

Alignment example.

User Input	Naive Formatting	Desired Formatting
$\begin{array}{rcl} a & = & b \\ \wedge & ccc >= & d \end{array}$	$\begin{array}{rcl} a & = & b \\ \wedge & ccc >= & d \end{array}$	$\begin{array}{rcl} a & = & b \\ \wedge & ccc \geq & d \end{array}$

should do. And thinking means specifying. When writing the pretty-printer for TLA+, I decided that instead of formatting formulas naively, it should align them the way the user intended (see the accompanying figure).

It is impossible to specify precisely what the user intended. My spec consisted of six alignment rules. One of them was:

If token t is a left-comment token, then it is left-comment aligned with its covering token.

where terms like *covering token* are defined precisely but informally. As I observed, this is usually not a good way to write a spec because it is hard to understand the consequences of a set of rules. So, while implementing the rules was easy, debugging them was not. But it was a lot easier to understand and debug six rules than 850 lines of code. (I added debugging statements to the code that reported what rules were being applied.) The resulting program does not always do the right thing; no program can when the right thing is subjective. However, it works much better, and took less time to write, than had I not written the spec. I recently enhanced the program to handle a particular kind of comment. The spec made this a simple task. Without the spec, I probably would have had to recode it from scratch. No matter how ill-defined a problem may be, a program to solve it has to do something. We will find a better solution by thinking about the problem and its solution, rather than just thinking about the code.

A related argument against specification is that the client often does not know what he wants, so we may as well just code as fast as we can so he can tell us what is wrong with the result. The blueprint metaphor easily refutes that argument.

The main goal of programmers

seems to be to produce software faster, so I should conclude by saying that writing specs will save you time. But I cannot. When performing any task, it is possible to save time and effort by doing a worse job. And the result of forcing a programmer to write a spec, when she is convinced that specs are a waste of time, is likely to be useless—just like a lot of documentation I have encountered. (Here is the description of the method *resetHighlightRange* in a class *TextEditor*: “Resets the highlighted range of this text editor.”)

To write useful specifications, you must want to produce good code—code that is easy to understand, works well, and has few errors. You must be sufficiently motivated to be willing to take the time to think and specify before you start coding. If you make the effort, specification can save time by catching design errors when they are easier to fix, before they are embedded in code. Formal specification can also allow you to make performance optimizations that you would otherwise not dare to try, because tools for checking your spec can give you confidence in their correctness.

There is nothing magical about specification. It will not eliminate all errors. It cannot catch coding errors; you will still have to test and debug to find them. (Language design and debugging tools have made great progress in catching coding errors, but they are not good for catching design errors.) And even a formal specification that has been proved to satisfy its required properties could be wrong if the requirements are incorrect. Thinking does not guarantee that you will not make mistakes. But not thinking guarantees that you will. ■

Leslie Lamport (lamport@microsoft.com) is a principal researcher at Microsoft Research and recipient of the 2013 ACM A.M. Turing Award.

Copyright held by author.

Article development led by [acmqueue](http://queue.acm.org)
queue.acm.org

**In the end, dynamic systems
are simply less secure.**

BY PAUL VIXIE

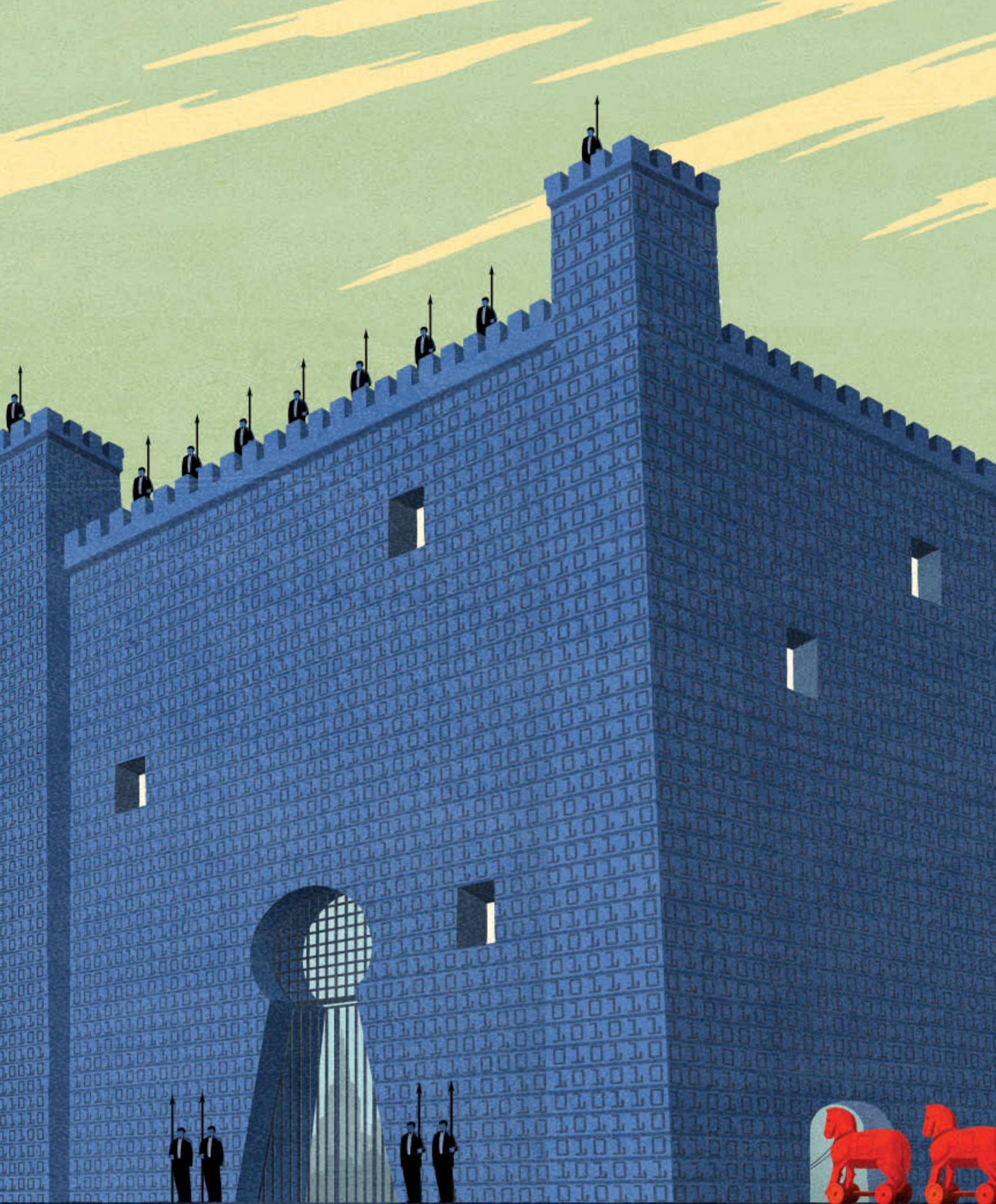
Go Static or Go Home

MOST CURRENT AND historic problems in computer and network security boil down to a single observation: *letting other people control our devices is bad for us.* At another time, I will explain what I mean by “other people” and “bad.” For the purpose of this article, I will focus entirely on what I mean by *control*. One way we lose control of our devices is to external distributed denial-of-service (DDoS) attacks, which fill a network with unwanted traffic, leaving no room for real (“wanted”) traffic. Other forms of DDoS are similar—an attack by the Low Orbit Ion Cannon (LOIC), for example, might not totally fill up a network, but it can keep a Web server so busy answering useless

attack requests the server cannot answer any useful customer requests. Either way, DDoS means outsiders are controlling our devices, and that is bad for us.

Surveillance, exfiltration, and other forms of privacy loss often take the form of malicious software or hardware (so, “malware”) that somehow gets into your devices, adding features like reading your address book or monitoring your keystrokes and reporting that information to outsiders. Malware providers often know more about our devices than we as users (or makers) do, especially if they have poisoned our supply chain. This means we sometimes use devices we do not consider to be programmable, but which actually are programmable by an outsider who knows of some vulnerability or secret handshake. Surveillance and exfiltration are merely examples of a device doing things its owner does not know about, would not like, and cannot control.

Because the Internet is a distributed system, it involves sending messages between devices such as computers and smartphones, each containing some hardware and some software. By far the most common way that malware is injected into these devices is by sending a message that is malformed in some deliberate way to exploit a bug or vulnerability in the receiving device’s hardware or software, such that something we thought of as data becomes code. Most defense mechanisms in devices that can receive messages from other devices prevent the promotion of data that is expected to contain text or graphics or maybe a spreadsheet to *code*, meaning instructions to the device telling it how to behave (or defining its features). The failed promise of anti-virus software was that malware could be detected using pattern matching. Today we use anti-virus tools to clean up infected systems, but we know we cannot count on detecting malware in time to prevent infection.



So we harden our devices, to try to keep data outside from becoming code inside. We shut off unnecessary services, we patch and update our operating systems, we use firewalls to control who can reach the services we cannot shut off, we cryptographically sign and verify our code, and we randomize the placement of objects in program memory so if data from the outside does somehow become code inside our devices, that code will guess wrong about where it landed and so fail to hurt us. We make some parts of our device memory data-only and other parts code-only, so a successful attack will put the outsider's data into a part of our device memory where code is not allowed to be executed.

We log accesses to our systems, hits on our firewalls, and flows on our networks, trying to calibrate the normal so as to highlight the abnormal. We buy subscriptions to network reputation systems so other devices known to be infected with malware cannot reach our services. We add CAPTCHAs to customer registration systems to keep botnets from creating fake accounts from which to attack us from inside our perimeters. We put every Internet-facing service into its own virtual machine so a successful attack will reach only a tiny subset of our enterprise.

Inviting the Trojan Horse Inside

And then, after all that spending on all that complexity for defense, some of us go on to install a Dynamic Content Management System (DCMS) as our public-facing Web server. This approach is like building a mighty walled city and then inviting the Trojan horse inside, or making Achilles invulnerable to harm except for his heel. WordPress and Drupal are examples of DCMSs, among hundreds. DCMSs have a good and necessary place in website management, but that place is not on the front lines where our code is exposed to data from the outside.

The attraction of a DCMS is that nontechnical editors can make changes or additions to a website, and those changes become visible to the public or to customers almost instantly. In the early days of the World Wide Web, websites were written in raw HTML using text editors



A Dynamic Content Management System is extremely dangerous—to the operators who use it.



on UNIX servers, which means, in the early days of the Web, all publication involved technical users who could cope with raw HTML inside a UNIX text editor. While I personally think of those as “the good old days,” I also confess the Web was, when controlled entirely by technical users, both less interesting and less productive than it is today. A DCMS is what enables the Web to fulfill the promise of the printing press—to make every person a potential publisher. Human society fails to thrive when the ability to speak to the public is restricted to the wealthy, to the powerful, or to the highly technical.

And yet, a DCMS is extremely dangerous—to the operators who use it. This is because of the incredible power and elasticity of the computer languages used to program DCMSs, and the power and elasticity of the DCMSs themselves. A DCMS gives us a chance to re-fight and often relose the war between data on the outside and code on the inside. Most of the computer languages used to write Web applications such as DCMSs contain a feature called *eval*, where programming instructions can be deliberately promoted from data to code at runtime. I realize that sounds insane, and it sort of is insane, but *eval* is merely another example of how *all power tools can kill*. In the right skilled hands, *eval* is a success-maker, but when it is left accessible to unskilled or malicious users, *eval* is a recipe for disaster. If you want to know how excited and pleased an attacker will be when they find a new way to get your code to *eval* their data, search the Web for “Little Bobby Tables.”

But even without *eval* in the underlying computer language used to program a DCMS or in the database used to manage that program's long-term data, such as student records, most DCMSs are internally data driven, meaning that DCMS software is often built like a robot that treats the website's content as a set of instructions to follow. To attack a DCMS by getting it to promote data to code, sometimes all that is needed is to add a specially formatted blog post or even a comment on an existing blog post. To defend a DCMS against this

kind of attack, what is needed is to audit every scrap of software used to program the DCMS, including the computer language interpreter; all code libraries, especially Open SSL; the operating system including its kernel, utilities, and compilers; the Web server software; and any third-party apps that have been installed alongside the DCMS. (Hint: this is ridiculous.)

Distributed Denial of Service

Let's rewind from remote code execution vulnerability (the promotion of outsider data into executable code) back to DDoS for a moment. Even if your DCMS is completely non-interactive, such that it never offers its users a chance to enter any input, the input data path for URLs and request environment variables has been carefully audited, and there is nothing like Bash installed on the same computer as the Web server, a DCMS is still a "kick me" sign for DDoS attacks. This is because every DCMS page view involves running a few tiny bits of software on your Web server, rather than just returning the contents of some files that were generated earlier. Executing code is quite fast on modern computers, but still far slower than returning the contents of pre-generated files. If someone is attacking a Web service with LOIC or any similar tool, they will need 1,000 times fewer attackers to exhaust a DCMS than to exhaust a static or file-based service.

Astute readers will note that my personal website is a DCMS. Instead of some lame defense like "the cobbler's children go shoeless," I will point out the attractions of a DCMS are so obvious that even I can see them—I do not like working on raw HTML using UNIX text editors when I do not have to, and my personal Web server is not a revenue source and contains no sensitive data. I do get DDoS'd from time to time, and I have to go in periodically and delete a lot of comment spam. The total cost of ownership is pretty low, and if your enterprise website is as unimportant as my personal website, then you should feel free to run a DCMS like I do. (Hint: wearing a "kick me" sign on your enterprise website may be bad for business.)

At work, our public-facing website is completely static. There is a Con-

tent Management System (CMS), but it is extremely technical—it requires the use of UNIX text editors, a version control utility called GIT, and knowledge of a language called Markdown. This frustrates our non-technical employees, including some members of our business team, but it means our Web server runs no code to render a Web object—it just returns files that were pre-generated using the Ikiwiki CMS. Bricolage is another example of a non-dynamic CMS but is friendlier to non-technical WYSIWYG users than something like Ikiwiki. Please note that nobody is DDoS-proof, no matter what their marketing literature or their annual report may say. We all live on an Internet that lacks any kind of admission control, so most low-investment attackers can trivially take out most high-investment defenders. However, we do have a choice about whether our website wears a "kick me" sign.

There is a hybrid model, which I will call *mostly static*, where all the style sheets, graphics, menus, and other objects that do not change between views and can be shared by many viewers are pre-generated and are served as files. The Web server executes no code on behalf of a viewer until that viewer has logged in, and even after that, most of the objects returned on each page view are static (from files). This is a little bit less safe than a *completely static* website, but it is a realistic compromise for many Web service operators. I say "less safe," because an attacker can register some accounts within the service in order to make their later attacks more effective. Mass account creation is a common task for botnets, and so most Web service operators who allow online registration try to protect their service using CAPTCHAs.

The mostly static model also works with Content Distribution Networks (CDNs) where the actual front end server that your viewers' Web browsers are connecting to is out in the cloud somewhere, operated by experts, and massively overprovisioned to cope with all but the highest-grade DDoS attacks. To make this possible, a website has to signal static objects such as graphics, style sheets, and JavaScript files are cacheable. This

tells the CDN provider that it can distribute those files across its network and return them many times to many viewers—and in case of a DDoS, many times to many attackers. Of course, once a user logs into the site, there will be some dynamic content, which is when the CDN will pass requests to the real Web server, and the DCMS will be exposed to outsider data again. This must never cease to be a cause for concern, vigilance, caution, and contingency planning.

As a hybrid almost-CDN model, a mostly static DCMS might be put behind a front-end Web proxy such as Squid or the *mod_proxy* feature of Apache. This will not protect your network against DDoS attacks as well as outsourcing to a CDN would do, but it can protect your DCMS's resources from exhaustion. Just note that any mostly static model (CDN or no CDN) will still fail to protect your DCMS code from exposure to outsider data. What this means for most of us in the security industry is that static is better than mostly static if the business purpose of the Web service can be met using a static publication model.

So if you are serious about running a Web-based service, don't put a "kick me" sign on it. Go static, or go home. ■

Related articles on queue.acm.org

Finding More Than One Worm in the Apple

Mike Bland

<http://queue.acm.org/detail.cfm?id=2620662>

Internal Access Controls

Geetanjali Sampemane

<http://queue.acm.org/detail.cfm?id=2697395>

DNS Complexity

Paul Vixie

<http://queue.acm.org/detail.cfm?id=1242499>

Paul Vixie is the CEO of Farsight Security. He previously served as president, chairman, and founder of ISC (Internet Systems Consortium); president of MAPS, PAIX, and MIBH; and CTO of Abovenet/MFN.

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

The perpetual motion of parallel performance.

BY NEIL J. GUNTHER, PAUL PUGLIA, AND KRISTOFER TOMASETTE

Hadoop Superlinear Scalability

“WE OFTEN SEE more than 100% speedup efficiency!” came the rejoinder to the innocent reminder that you cannot have more than 100% of anything. This was just the first volley from software engineers during a presentation on how to quantify computer-system scalability in terms of the speedup metric. In different venues, on subsequent occasions, that retort seemed to grow into a veritable chorus that not only was superlinear speedup commonly observed, but also the model used to quantify scalability for the past 20 years—Universal Scalability Law (USL)—failed when applied to superlinear speedup data.

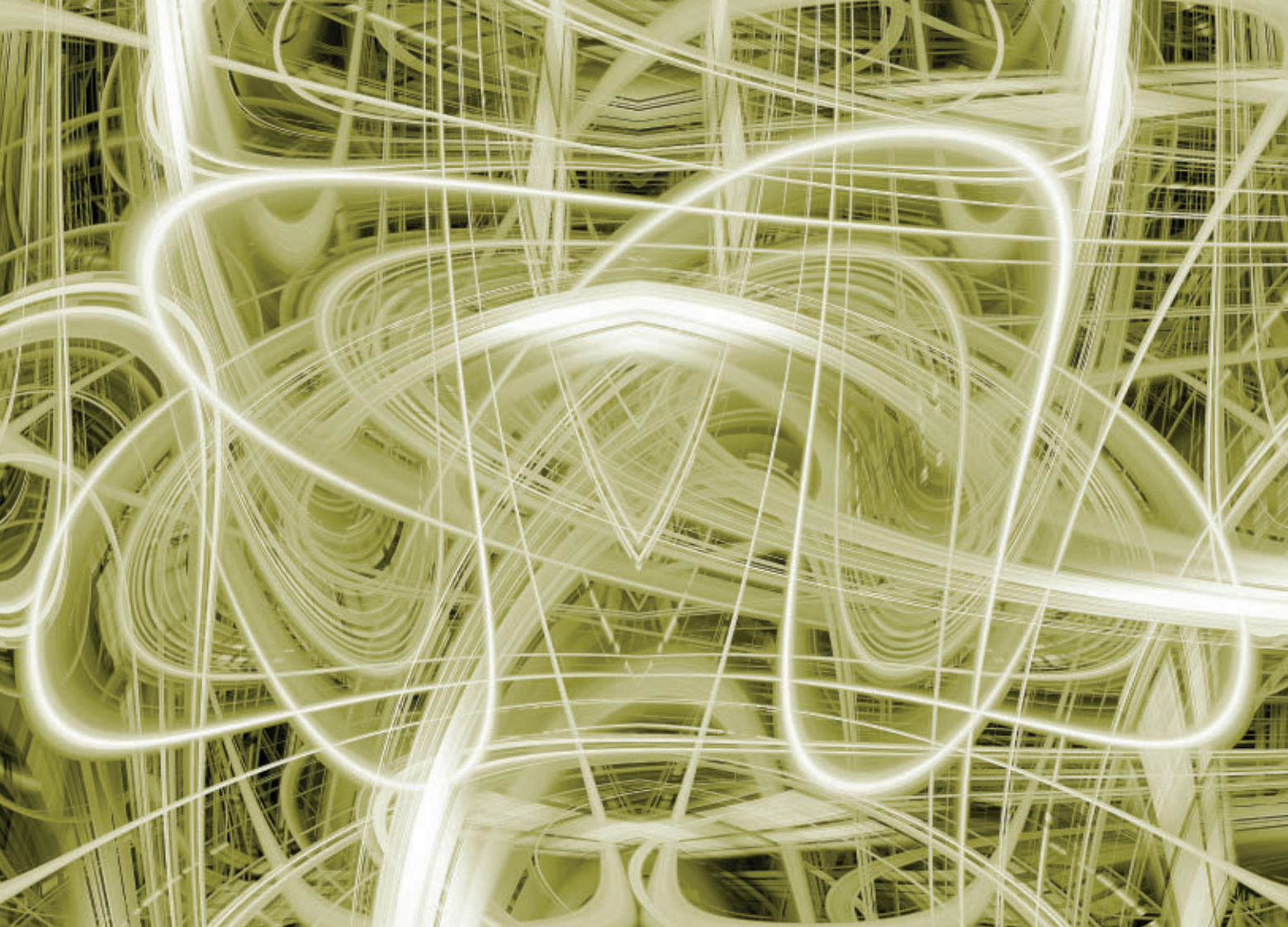
Indeed, superlinear speedup is a bona fide phenomenon that can be expected to appear more frequently in practice as new applications are deployed onto distributed architectures. As demonstrated here using Hadoop MapReduce, however, the USL is not only capable of accommodating superlinear speedup in a surprisingly simple way, it also reveals that superlinearity, although alluring, is as illusory as perpetual motion.

To elaborate, Figure 1 shows conceptually that linear speedup (the dashed line) is the best you can ordinarily expect to achieve when scaling an application. Linear means you get equal bang for your capacity buck because the available capacity is being consumed at 100% efficiency. More commonly, however, some of that capacity is consumed by various forms of overhead (red area). That corresponds to a growing loss of available capacity for the application, so it scales in a sublinear fashion (red curve). Superlinear speedup (blue curve), on the other hand, seems to arise from some kind of hidden capacity boost (green area).

As we will demonstrate, superlinearity is a genuinely measurable effect,^{4,12,14,19,20,21–23} so it is important to understand exactly what it represents in order to address it when sizing distributed systems for scalability. As far as we are aware, this has not been done before.

Measurability notwithstanding, superlinearity is reminiscent of *perpetuum mobile* claims. What makes a perpetual motion machine attractive is its supposed ability to produce more work or energy than it consumes. In the case of computer performance, superlinearity is tantamount to speedup that exceeds the computer capacity available to support it. More importantly for this discussion, when it comes to perpetual motion machines, the difficult part is not deciding if the claim violates the conservation of energy law; the difficult part is *debugging* the machine to find the flaw in the logic. Sometimes that endeavor can even prove fatal.⁵

If, *prima facie*, superlinearity is akin to perpetual motion, why would some software engineers be proclaiming its ubiquity rather than debugging it? That kind of exuberance comes from an overabundance of trust in performance data. To be fair, that misplaced trust likely derives from the way performance data is typically presented without any indication of measurement error. No open source or commercial performance tools of which we are aware display measurement errors,



even though all measurements contain errors. Put simply, all measurements are “wrong” by definition: the only question is, how much “wrongness” can be tolerated? That question cannot be answered without quantifying measurement error. (Later in this article, Table 2 quantifies Hadoop measurement errors.)

In addition to determining measurement errors, all performance data should be assessed within the context of a validation method. One such method is a performance model. In the context of superlinear speedup, the USL^{6-11,17,18} fulfills that role in a relatively simple way.

Universal Scalability Model

To quantify scalability more formally, we first define the empirical speedup metric in equation 1:

$$S_p = \frac{T_1}{T_p} \quad (1)$$

where T_p is the measured runtime on $p = 1, 2, 3, \dots$ processors or cluster nodes.¹⁴ Since the multinode

runtime T_p is expected to be shorter than the single-node runtime T_1 , the speedup is generally a discrete concave function of p . The following special cases can be identified.

► **Linear scalability.** If $T_p = T_1/p$ for each cluster configuration, then the speedup will have values $S_p = 1, 2, 3, \dots$ for each p , respectively. The speedup function exhibits linear scalability (the dashed line in Figure 1).

► **Sublinear scalability.** If $T_p > T_1/p$ for each cluster configuration, then successive speedup values will be *inferior* (red curve) to the linear scalability bound in Figure 1. For example, if $p = 2$ and $T_2 = 3T_1/4$, then $S_2 = 1.33$. Since it is less than $S_2 = 2$, the speedup is sublinear. The red curve is the most common form of scalability observed on both monolithic and distributed systems.

► **Superlinear scalability.** If $T_p < T_1/p$, then successive speedup values will be superior (blue curve) to the linear bound in Figure 1. For example, if $p = 2$ and $T_2 = T_1/3$, then $S_2 = 3$, which is greater than linear speedup.

The scalability of any computer system can be validated by comparing the measured speedup in equation 1 with the theoretically expected speedup, defined here.

Components of scalability. Scalability, treated as an aggregation of computer hardware and software, can be thought of as resulting from several physical factors:

1. Ideal parallelism or maximal concurrency.
2. Contention for shared resources.
3. Saturation resulting from the primary bottleneck resource.
4. Exchange of data between nonlocal resources to reach consistency or data coherency.

This does not yet take superlinearity into consideration. The individual effect of these factors on scalability, as measured by the speedup metric in equation 1, is shown schematically in Figure 2.

Each of these scaling effects can also be represented as separate terms in an analytic performance model: the

USL. In our usual notation,^{7,8} we write the theoretical speedup in equation 2 as:

$$S_p = \frac{p}{1 + \sigma p + \kappa p(p-1)} \quad (2)$$

where the coefficient σ represents the degree of contention in the system, and the coefficient κ represents the incoherency in distributed data.

The *contention* term in equation 2 grows linearly with the number of cluster nodes, p , since it represents the cost of waiting for a shared resource such as message queuing. The coherency term grows quadratically with p because it represents the cost of making distributed data consistent (or coherent) via a pairwise exchange between distributed resources (for example, processor caches).

Interpreting the coefficients. If $\sigma = 0$ and $\kappa = 0$, then the speedup simply reduces to $S_p = p$, which corresponds to Figure 2a. If $\sigma > 0$, the speedup starts to

fall away from linear (Figure 2b), even when the node configuration is relatively small. As the number of nodes continues to grow, the speedup approaches the ceiling, $S_{ceiling} = \sigma^{-1}$, indicated by the horizontal dashed line in Figure 2c. The two triangles in Figure 2c indicate this is a region of diminishing returns, since both triangles have the same width but the right triangle has less vertical gain than the left triangle.

If $\kappa > 0$, the speedup will eventually degrade like p^{-1} . The continuous scalability curve must therefore pass through a maximum or peak value, as in Figure 2d. Although both triangles are congruent, the one on the right side of the peak is reversed, indicating the slope has become negative—a region of negative returns.

From a mathematical perspective, the USL is a parametric model based on rational functions,⁷ and one could imagine continuing to add successive polynomial terms in p to the denomi-

nator of equation 2, each with its attendant coefficient. For $\kappa > 0$, however, a maximum exists, and there is usually little virtue in describing analytically how scalability degrades beyond that point. The preferred goal is to remove the peak altogether, if possible—hence the name *universal*.

The central idea is to match the measured speedup in equation 1 with the USL defined in equation 2. For a given node configuration p , this can be achieved only by adjusting the σ and κ coefficients. In practice, this is accomplished using nonlinear statistical regression.^{8,17} (The scalability of Varnish, Memcached, and Zookeeper applications are discussed in the *ACM Queue* version of this article).

Hadoop Terasort in the Cloud

To explore superlinearity in a controlled environment, we used a well-known workload, the TeraSort benchmark,^{15,16} running on the Hadoop MapReduce framework.^{3,24} Instead of using a physical cluster, however, we installed it on Amazon Web Services (AWS) to provide the flexibility of reconfiguring a sufficiently large number of nodes, as well as the ability to run multiple experiments in parallel at a fraction of the cost of the corresponding physical system.

Hadoop framework overview. This discussion of superlinear speedup in TeraSort requires some familiarity with the Hadoop framework and its terminology.²⁴ In particular, this section provides a high-level overview with the primary focus on just those Hadoop components that pertain to the later performance analysis.

The Hadoop framework is designed to facilitate writing large-scale, data-

Figure 1. Qualitative comparison of sublinear, linear, and superlinear speedup scalability.

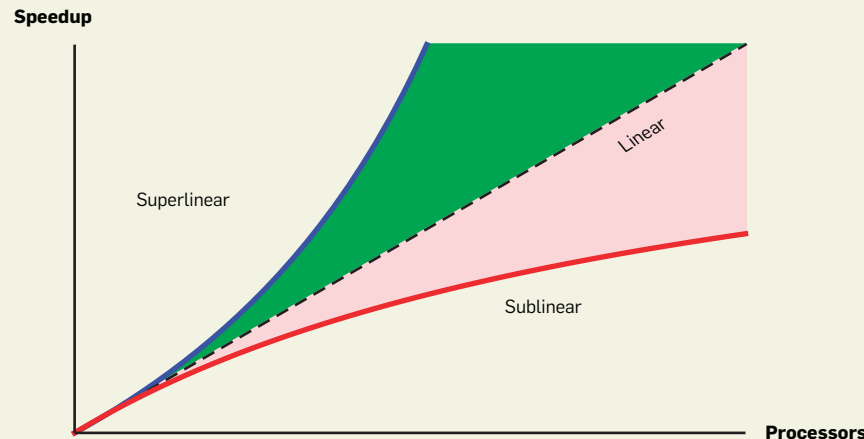
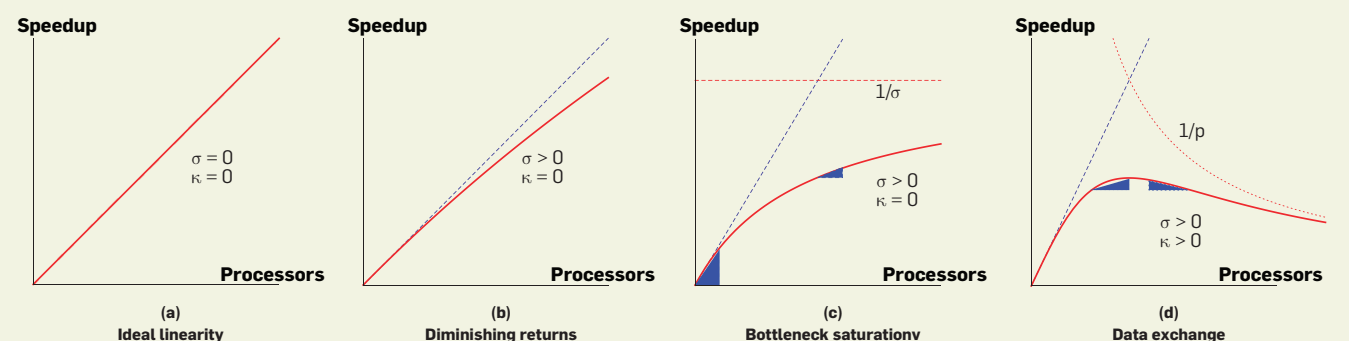


Figure 2. How the USL model in equation 2 characterizes scalability.



intensive, distributed applications that can run on a multinode cluster of commodity hardware in a reliable, fault-tolerant fashion. This is achieved by providing application developers with two programming libraries:

- *MapReduce*, a distributed processing library that enables applications to be written for easy adaptation to parallel execution by decomposing the entire job into a set of independent tasks.

- *Hadoop Distributed File System* (HDFS), which allows data to be stored on any node but remain accessible by any task in the Hadoop cluster. An application written using the MapReduce library is organized as a set of independent tasks that can be executed in parallel. These tasks fall into two classes:

- *Map task*. The function of the Map task is to take a slice of the entire input dataset and transform it into key-value pairs, commonly denoted by $\langle \text{key}, \text{value} \rangle$ in the context of MapReduce. See the detailed Map-tasks dataflow in

Node 1 of Figure 3 where the Map task is represented schematically as a procedure $\text{Map}(k, v)$. Besides performing this transform, the Map task also sorts the data by key and stores the sorted $\langle k, v \rangle$ objects so they can easily be exchanged with a Reduce task.

- *Reduce task*. The function of the Reduce task is to collect all the $\langle k, v \rangle$ objects for a specific key and transform them into a new $\langle k, v \rangle$ object, where the value of the key is the specific key and whose value is a list $[v_1, v_2, \dots]$ of all the values that are $\langle k, [v_1, v_2, \dots] \rangle$ objects whose key is the specific key across the entire input data set. Node 1 of Figure 3 shows the detailed Reduce-task dataflow.

A MapReduce application processes its input dataset using the following workflow. On startup, the application creates and schedules one Map task per slice of the input dataset, as well as creating a user-defined number of Reduce tasks. These Map tasks then

work in parallel on each slice of the input data, effectively sorting and partitioning it into a set of files where all the $\langle k, v \rangle$ objects that have equal key values are grouped. Once all the Map tasks have completed, the Reduce tasks are signaled to start reading the partitions to transform and combine these intermediate data into new $\langle k, [v_1, v_2, \dots] \rangle$ objects. This process is referred to as *shuffle exchange*, shown schematically in Figure 3 as arrows spanning physical nodes 1, 2, ..., p .

To facilitate running the application in a distributed fashion, the MapReduce library provides a distributed execution server composed of a central service called the JobTracker and a number of slave services called TaskTrackers.²⁴ The JobTracker is responsible for scheduling and transferring tasks to the TaskTrackers residing on each cluster node. The JobTracker can also detect and restart tasks that might fail. It provides a level of fault toler-

Figure 3. Hadoop MapReduce dataflow with Node 1 expanded to show tasks detail.

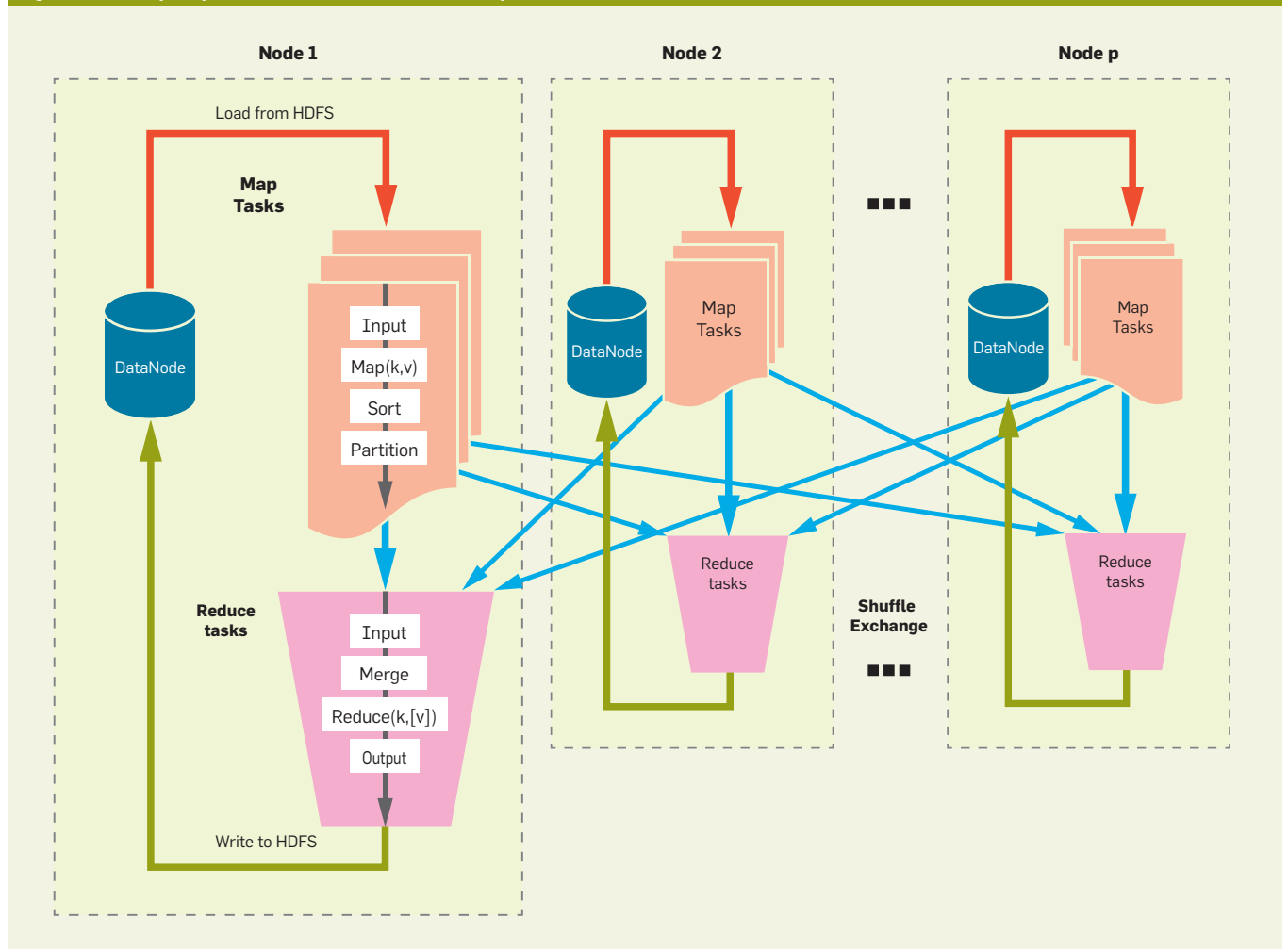


Table 1. Amazon EC2 instance configurations.

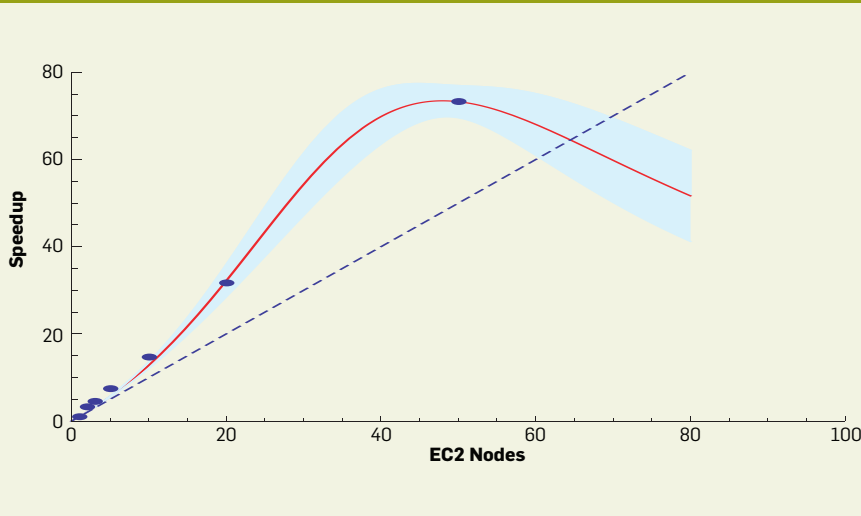
	Instance Type	Optimized for	Processor Arch.	vCPU number	Memory (GiB)	Instance Storage (GB)	Network Performance
BigMem	m2.2xlarge	Memory	64-bit	4	34.2	1 x 850	Moderate
BigDisk	c1.xlarge	Compute	64-bit	8	7	4 x 420	High

Figure 4. Bash script used to record Terasort elapse times.

```

BEFORE_SORT='date +%s%3N'
hadoop jar $HADOOP_MAPRED_HOME/hadoop-examples.jar terasort /user/hduser/
terasort-input
/user/hduser/terasort-output
AFTER_SORT='date +%s%3N'
SORT_TIME='expr $AFTER_SORT - $BEFORE_SORT' echo "$CLUSTER_SIZE, $SORT_TIME"
>> sort_time

```

Figure 5. USL analysis of superlinear speedup for $p \leq 50$ BigMem nodes.

ance. The user interacts with the Hadoop framework via a JobClient component such as TeraSort that provides monitoring and control of the MapReduce job.

To support the execution of MapReduce tasks, the Hadoop framework includes HDFS, which is implemented as a storage cluster using a master-slave architecture. It provides a reliable distributed file service that allows Hadoop applications to read and write very large data files at high throughput of fixed-sized blocks (128MB in TeraSort³) across the cluster. The master node in a HDFS cluster is the NameNode, which is responsible for regulating client access to files, as well as managing the file-system namespace by mapping file blocks to its storage location, which can reside on the DataNodes (that

is, slave nodes to the NameNode). A key feature of HDFS is its built-in resilience to node disk failure, accomplished by replicating blocks across multiple DataNodes. The default replication factor is three, but this is set to one in TeraSort.

It is noteworthy the shuffle-exchange process depicted in Figure 3 involves the interaction between Map and Reduce tasks, which, in general, causes data to be ‘Reduced’ on different physical nodes. Since this exchange occurs between MapReduce pairs, it scales quadratically with the number of cluster nodes, and that corresponds precisely to the USL coherency term, $p(p-1)$, in equation 2 (Figure 2d). This point will be important for the later performance analysis of superlinear speedup. Moreover, although sorting represents a worst-

case MapReduce workload, similar coherency phases will occur with different magnitudes in different Hadoop applications. The actual magnitude of the physical coherency effect is reflected in the value of the κ coefficient that results from USL analysis of Hadoop performance data.

Running TeraSort on AWS. TeraSort is a synthetic workload that has been used recently to benchmark the performance of Hadoop MapReduce^{15,16} by measuring the time taken to sort 1TB of randomly generated data. A separate program called TeraGen generates the input data, consisting of 100-byte records with the first 10 bytes used as a key.

The scripts for setting TeraSort up on a Hadoop cluster are readily available. The performance goal here was to use TeraSort to examine the phenomenon of superlinear scalability, not to tune the cluster to produce the shortest runtimes as demanded by competitive benchmarking.^{15,16}

Amazon’s Elastic Compute Cloud (EC2) provides rapid and cheap provisioning of clusters with various instance types and sizes (for example, those in Table 1). To keep the time and cost of running multiple experiments manageable, data generation was limited to 100GB and the cluster configurations kept to less than 200 EC2 nodes using local instance storage rather than Elastic Block Storage.

EC2 instance types m2.2xlarge and c1.xlarge are distinguished by the former having five times more memory, but only one hard disk, half the number of cores, and higher network latencies, whereas the latter has four hard disks and lower network latency. Rather than adhering to the clumsy Amazon nomenclature, we refer to them here as BigMem and BigDisk, respectively. These designations emphasize the key capacity difference that will turn out to be important for the later performance analysis.

Apache Whirr¹ is a set of Java libraries for running cloud services that supports Amazon EC2. We used it together with custom bash scripts to: specify cluster size and instance type; bootstrap the EC2 cluster; install Hadoop; prepare and run TeraSort; and collect performance metrics.

Whirr was configured to create a cluster made of EC2 instances running

Linux CentOS 5.4 with the Cloudera CDH 4.7.0 distribution of Hadoop 1.0 installed.³ Included in that distribution is the `Hadoop-examples.jar` file that contains the code for both the TeraGen and TeraSort MapReduce jobs. Whirr can read the desired configuration from a properties file, as well as receiving properties passed from the command line. This allowed permanent storage of the parameters that did not change (for example, the operating system version and Amazon credentials).

Three sets of performance metrics were gathered:

- ▶ The elapsed time for the TeraSort job (excluding the TeraGen job).
- ▶ Hadoop-generated job data files.
- ▶ Linux performance metrics.

Of these, the most important metric was the elapsed time, which was recorded using the Posix time stamp in milliseconds (since EC2 hardware supports it) via the shell command illustrated in Figure 4.

Runtime performance metrics (for example, memory usage, disk IO rates, and processor utilization) were captured from each EC2 node using the resident Linux performance tools `uptime`, `vmstat`, and `iostat`. The performance data was parsed and appended to a file every two seconds.

A sign of perpetual motion. Figure 5 shows the TeraSort speedup data (dots) together with the fitted USL scalability curve (blue). The linear bound (dashed line) is included for reference. That the speedup data lies on or above the linear bound provides immediate visual evidence that scalability is indeed superlinear. Rather than a linear fit,²¹ the USL regression curve exhibits a convex trend near the origin that is consistent with the generic superlinear profile in Figure 1.

The entirely unexpected outcome is that the USL contention coefficient develops a *negative* value: $\sigma = -0.0288$. This result also contradicts the assertion⁸ that both σ and κ must be positive

for physical consistency—the likely source of the criticism that the USL failed with superlinear speedup data.

As explained earlier, a positive value of σ is associated with contention for shared resources. For example, the same processor that executes user-level tasks may also need to accommodate operating-system tasks such as IO requests. The processor capacity is consumed by work other than the application itself. Therefore, the application throughput is less than the expected linear bang for the capacity buck.

Capacity consumption ($\sigma > 0$) accounts for the sublinear scalability component in Figure 2b. Conversely, $\sigma < 0$ can be identified with some kind of capacity boost. This interpretation will be explained shortly.

Additionally, the (positive) coherency coefficient $\kappa = 0.000447$ means there must be a peak value in the

speedup, which the USL predicts as $S_{max} = 73.48$, occurring at $p = 48$ nodes. More significantly, it also means the USL curve must cross the linear bound and enter the payback region shown in Figure 6.

The USL model predicts this crossover from the superlinear region to the payback region must occur for the following reason. Although the magnitude of σ is small, it is also multiplied by $(p - 1)$ in equation 2. Therefore, as the number of nodes increases, the difference, $1 - \sigma(p - 1)$, in the denominator of equation 2 becomes progressively smaller such that S_p is eventually dominated by the coherency term, $\kappa p(p - 1)$.

Figure 7 includes additional speedup measurements (squares). The fitted USL coefficients are now significantly smaller than those in Figure 5. The maximum speedup, S_{max} , therefore is about 30% higher than predicted with the data

Figure 6. Superlinearity and its associated payback region (see Figure 1).

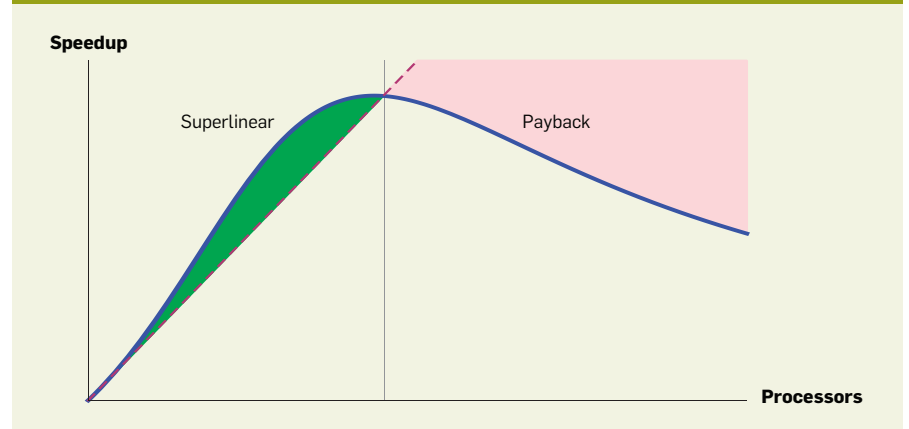


Figure 7. USL analysis of $p \leq 150$ BigMem nodes (solid blue curve) with Figure 4 (dashed blue curve) inset for comparison.

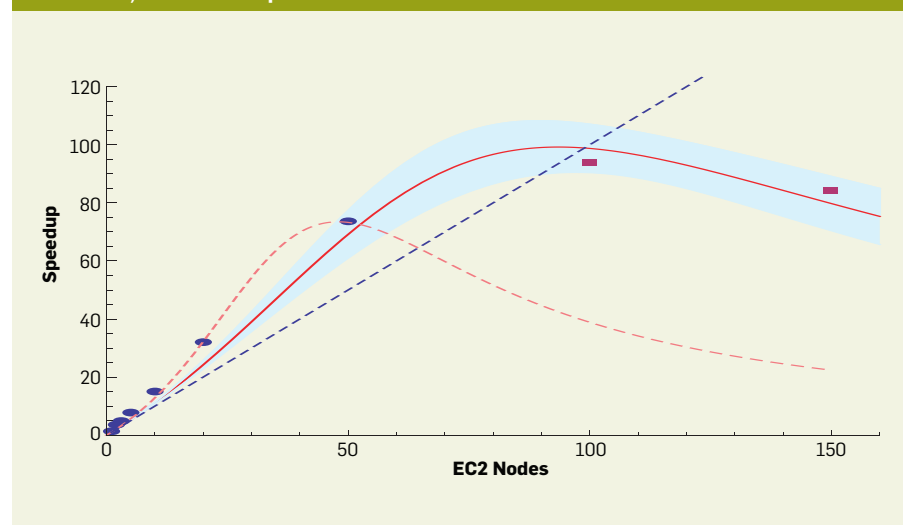
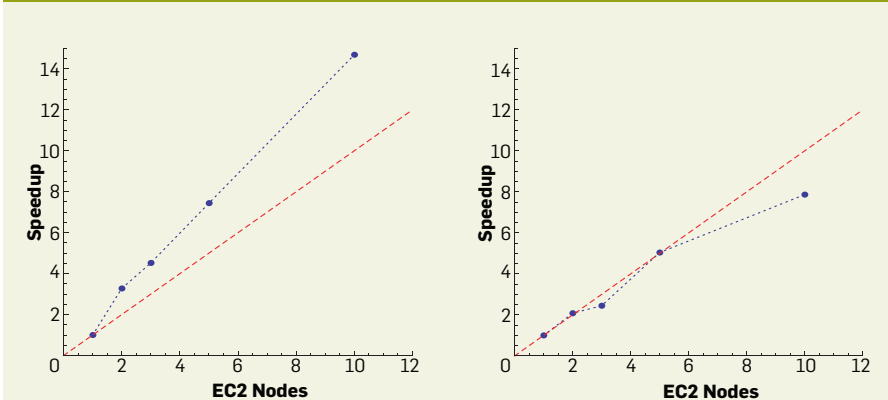


Table 2. Runtime error analysis.

T1	=	13057	±	606	seconds (r.e. 5%)
T2	=	6347	±	541	seconds (r.e. 9%)
T3	=	4444	±	396	seconds (r.e. 9%)
T5	=	2065	±	147	seconds (r.e. 7%)
T10	=	893	±	27	seconds (r.e. 3%)

Figure 8. Hadoop TeraSort superlinearity is eliminated by increasing nodal disk IO bandwidth.



in Figure 5 and occurs at $p = 95$ nodes. The measured values of the speedup differ from the original USL prediction, not because the USL is wrong but because there is now more information available than previously. Moreover, this confirms the key USL prediction that superlinear speedup reaches a maximum value and then rapidly declines into the payback region.

Based on USL analysis, the scalability curve is expected to cross the linear bound at p_x nodes given by equation 3:

$$p_x = \left\lceil \frac{|\sigma|}{K} \right\rceil \quad (3)$$

For the dashed curve in Figure 7, the crossover occurs at $p_x = 65$ nodes, whereas for the solid curve it occurs at $p_x = 99$ nodes. Like predicting S_{max} , the difference in the two p_x predictions comes from the difference in the amount of information contained in the two sets of measurements.

Hunting the Superlinear Snark

After the TeraSort data was validated against the USL model, a deeper performance analysis was needed to determine the cause of superlinearity. Let's start with a closer examination of the actual runtime measurements for each EC2 cluster configuration.

Runtime data analysis. To make a statistical determination of the error in the runtime measurements, we performed some runs with a dozen repetitions per node configuration. From that sample size a reasonable estimate of the uncertainty can be calculated based on the standard error, or the relative error, which is more intuitive.

For each of the runtimes in Table 2, the number before the $\pm$ sign is the sample mean, while the error term following the $\pm$ sign is derived from the sample variance. The relative error (r.e.) is the ratio of the standard error to the mean value reported as a percentage.

What is immediately evident from this numerical analysis is the significant variation in the relative errors with a range from 3%, which is nominal, to 9%, which likely warrants further attention. This variation in the measurement error does not mean the measurement technique is unreliable; rather, it means there is a higher degree of dispersion in the runtime data for reasons that cannot be discerned at this level of analysis.

Nor is this variation in runtimes peculiar to our EC2 measurements. The Yahoo TeraSort benchmark team also noted significant variations in their execution times, although they did not quantify them. *"Although I had the 910 nodes mostly to myself, the network core was shared with another active 2000 node cluster, so the times varied a lot depending on the other activity."*¹⁵

Some of the Yahoo team's sources of variability may differ from ours (for example, the 10x larger cluster size is likely responsible for some of the Yahoo variation). *"Note that in any large cluster and distributed application, there are a lot of moving pieces and thus we have seen a wide variation in execution times."*¹⁶

A surprising hypothesis. The physical cluster configuration used by the Yahoo benchmark team comprised nodes with two quad-core Xeon processors (that is, a total of eight cores

per node) and four SATA disks.¹⁶ This is very similar to the BigDisk EC2 configuration in Table 1. We therefore repeated the TeraSort scalability measurements on the BigDisk cluster. The results for $p = 2, 3, 5$, and 10 clusters are compared in Figure 8.

Consistent with Figure 5, BigMem speedup values in Figure 8a are superlinear, whereas the BigDisk nodes in Figure 8b unexpectedly exhibit speedup values that are either linear or sublinear. The superlinear effect has essentially been eliminated by increasing the number of local spindles from one to four per cluster node. In other words, increasing nodal IO bandwidth leads to the counterintuitive result that scalability is degraded from superlinear to sublinear.

In an attempt to explain why the superlinear effect has diminished, we formed a working hypothesis by identifying the key performance differences between BigMem and BigDisk.

BigMem has the larger memory configuration, which possibly provides more CentOS buffer caching for the TeraSort data, and that could be thought of as being the source of the capacity boost associated with the negative USL contention coefficient. Incremental memory growth in proportion to cluster size is a common explanation for superlinear speedup.^{4,14} Increasing memory size, however, is probably not the source of the capacity boost in Hadoop TeraSort. If the buffer cache fills to the point where it needs to be written to disk, it will take longer because there is only a single local disk per node on BigMem. A single-disk DataNode in Figure 3 implies all disk IO is serialized. In this sense, when disk writes (including replications) occur, TeraSort is IO bound—most particularly in the single-node case. As the cluster configuration gets larger, this latent IO constraint becomes less severe since the amount of data per node that must be written to disk is reduced in proportion to the number of nodes. Successive cluster sizes therefore exhibit runtimes that are shorter than the single-node case, and that results in the superlinear speedup values shown in Figure 8a.

Conversely, although BigDisk has a smaller amount of physical memory per node, it has quad disks per

DataNode, which means each node has greater disk bandwidth to accommodate more concurrent IOs. TeraSort is therefore far less likely to become IO bound. Since there is no latent single-node IO constraint, there is no capacity boost at play. As a result, the speedup values are more orthodox and fall into the sublinear region of Figure 8b.

Note that since the Yahoo benchmark team used a cluster configuration with four SATA disks per node, they probably did not observe any superlinear effects. Moreover, they were focused on measuring elapsed times, not speedup, for the benchmark competition, so superlinearity would have been observable only as execution times T_p falling faster than p^{-1} .

Console stack traces. The next step was to validate the IO bottleneck hypothesis in terms of Hadoop metrics collected during each run. While TeraSort was running on BigMem, task failures were observed in the Hadoop JobClient console that communicates with the Hadoop JobTracker. The following is an abbreviated form of a failed task status message with the salient identifiers shown in bold in Figure 9.

Since the TeraSort job continued and all tasks ultimately completed successfully, we initially discounted these failure reports. Later, with the IO bottleneck hypothesis in mind, we realized these failures seemed to occur only during the Reduce phase. Simultaneously, the Reduce task %Complete value decreased immediately when a failure appeared in the console. In other words, progress of that Reduce task became retrograde. Moreover, given that the failure in the stack trace involved the Java class `DFSOutputStream`, we surmised the error was occurring while attempting to write to HDFS. This suggested examining the server-side Hadoop logs to establish the reason why the Reduce failures are associated with HDFS writes.

Hadoop log analysis. Searching the Hadoop cluster logs for the same failed `TASK_ATTEMPT_ID`, initially seen in the JobClient logs, revealed the corresponding record as shown in Figure 10.

This record indicates the Reduce task actually failed on the Hadoop

cluster, as opposed to the JobClient. Since the failure occurred during the invocation of `DFSOutputStream`, it also suggests there was an issue while physically writing data to HDFS.

Furthermore, a subsequent record in the log with the same task ID, as shown in Figure 11, had a newer `TASK_`

`ATTEMPT_ID` (namely, a trailing 1 instead of a trailing 0) that was successful.

This log analysis suggests if a Reduce task fails to complete its current write operation to disk, it must start over by rewriting that same data until it is successful. In fact, there may be multiple failures and retries (see Table

Figure 9. Failed Reduce task as seen in the Hadoop job-client console.

```
14/10/01 21:53:41 INFO mapred.JobClient: Task Id :
attempt_201410011835_0002_r_000000_0,
Status : FAILED java.io.IOException: All datanodes 10.16.132.16:50010 are bad.
Aborting . . .
. . .
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.
java:463)
```

Figure 10. Hadoop cluster log message corresponding to Figure 9.

```
ReduceAttempt TASK_TYPE="REDUCE" TASKID="task_201410011835_0002_r_000000"
TASK_ATTEMPT_ID="attempt_201410011835_0002_r_000000_0" TASK_STATUS="FAILED"
FINISH_TIME="1412214818818" HOSTNAME="ip-10-16-132-16.ec2.internal"
ERROR="java.io.IOException: All datanodes 10.16.132.16:50010 are bad. Aborting
. . .
. . .
at org.apache.hadoop.hdfs.DFSOutputStream$DataStreamer.run(DFSOutputStream.
java:463)
```

Figure 11. Successful retry of failed Reduce task.

```
ReduceAttempt TASK_TYPE="REDUCE" TASKID="task_201410011835_0002_r_000000"
TASK_ATTEMPT_ID=
"attempt_201410011835_0002_r_000000_1" TASK_STATUS= "SUCCESS"
```

Figure 12. Origin of Reduce task failure as seen in the Hadoop log.

```
ReduceAttempt TASK_TYPE="REDUCE" ... TASK_STATUS="FAILED" . . .
ERROR="java.io.IOException: All datanodes are bad. Aborting . . .
. . .
.setupPipelineForAppendOrRecovery(DFSOutputStream.java:1000)
```

Table 3. Single-node BigMem metrics extracted from Hadoop log.

Job ID	Finished Maps	Failed Maps	Finished Reduces	Failed Reduces	Job Runtime
1	840	0	3	0	9608794
2	840	0	3	2	12167730
3	840	0	3	0	10635819
4	840	0	3	1	11991345
5	840	0	3	0	11225591
6	840	0	3	2	12907706
7	840	0	3	2	12779129
8	840	0	3	3	13800002
9	840	0	3	3	14645896
10	840	0	3	4	15741466
11	840	0	3	2	14536452
12	840	0	3	3	16645014

3). The potential difference in runtime resulting from Reduce retries is obscured by the aforementioned variation in runtime measurements, which is also on the order of 10%.

Table 3 shows 12 rows corresponding to 12 parallel TeraSort jobs, each running on its own BigMem single-node cluster. A set of metrics indicating how each of the runs executed is stored in the Hadoop job-history log and extracted using Hadoop log tools.¹³

The 840 Map tasks are determined by the TeraSort job partitioning 100 (binary) GB of data into 128 (decimal) MB HDFS blocks. No Map failures occurred. The number of Reduce tasks was set to three per cluster node. The number of failed Reduce tasks varied randomly between none and four. In comparison, there were no Reduce failures for the corresponding BigDisk case.

The average runtime for Hadoop jobs was 13057078.67 ms, shown as T_1 in Table 2. Additional statistical analysis reveals a strong correlation between the number of Reduce task retries and longer runtimes. Recalling the definition of speedup, if the mean single-node runtime, T_1 , is longer than successive values of pT_p , then the speedup will be superlinear.

Whence reduce fails? The number

of failed Reduces in Table 3 indicates that a write failure in the Reduce task causes it to retry the write operation—possibly multiple times. In addition, failed Reduce tasks tend to incur longer runtimes as a consequence of those additional retries. The only outstanding question is, what causes the writes to fail in the first place? We already know that write operations are involved during a failure, and that suggests examining the HDFS interface.

Returning to the earlier failed Reduce stack trace, closer scrutiny reveals the following lines, with important key words shown in bold in Figure 12.

The “All datanodes are bad” Java IOException means the HDFS DataNode pipeline in Figure 13 has reached a state where the `setupPipelineForAppendOrRecovery` method, on the `DFSOutputStream` Java class, cannot recover the write operation, and the Reduce task fails to complete.

When the pipeline is unhindered, a Reduce task makes a call into the `HDFSClient`, which then initiates the creation of a HDFS DataNode pipeline. The `HDFSClient` opens a `DFSOutputStream` and readies it for writing (1. Write in Figure 13) by allocating a HDFS data block on a DataNode. The `DFSOutputStream` then breaks the data stream into smaller packets

of data. Before it transmits each data packet to be written by a DataNode (2. Write packet), it pushes a copy of that packet onto a queue. The `DFSOutputStream` keeps that packet in the queue until it receives an acknowledgment (3. ACK packet) from each DataNode that the write operation completed successfully.

When an exception is thrown (for example, in the stack trace) the `DFSOutputStream` attempts to remedy the situation by reprocessing the packets to complete the HDFS write. The `DFSOutputStream` can make additional remediation attempts up to one less than the replication factor. In the case of TeraSort, however, since the replication factor is set to one, the lack of a single HDFS packet acknowledgment will cause the entire `DFSOutputStream` write operation to fail.

The `DFSOutputStream` endeavors to process its data in an unfettered way, assuming the DataNodes will be able to keep up and respond with acknowledgments. If, however, the underlying IO subsystem on a DataNode cannot keep up with this demand, an outstanding packet can go unacknowledged for too long. Since there is only a single replication in the case of TeraSort, no remediation is undertaken. Instead, the `DFSOutputStream` immediately regards the outstanding write packet to be AWOL and throws an exception that propagates back up to the Reduce task in Figure 13.

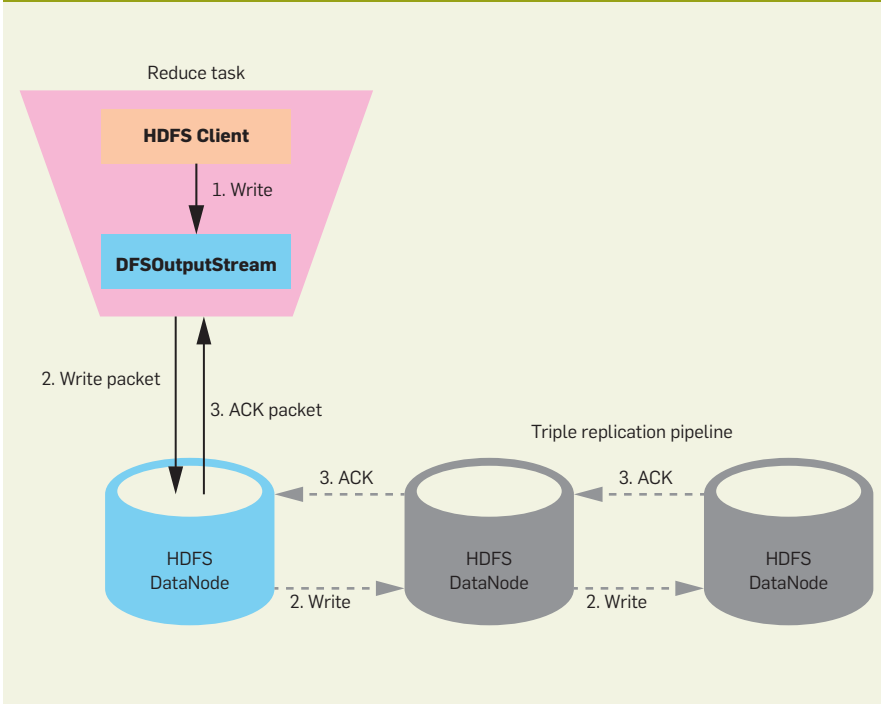
Since the Reduce task does not know how to handle this IO exception, it completes with a `TASK_STATUS=“FAILED”`. The MapReduce framework will eventually retry the entire Reduce task, possibly more than once (see Table 3), and that will be reflected in a stretched T_1 value that is ultimately responsible for the observed superlinear speedup.

This operational insight into Reduce failures can be used to construct a list of simple tactics to avoid runtime stretching.

1. Resize the buffer cache.
2. Tune kernel parameters to increase IO throughput.
3. Reconfigure Hadoop default timeouts.

If maintaining a BigMem-type cluster is dictated by nonengineering requirements (for example, budgetary

Figure 13. HDFS DataNode pipeline showing single replication (blue) and default triple replication (blue and gray).



constraints), then any of these steps could be helpful in ameliorating super-linear effects.

Conclusion

The large number of controlled measurements performed by running Hadoop TeraSort on Amazon EC2 exposed the underlying cause of super-linearity that would otherwise be difficult to resolve in the field. Fitting our speedup data to the USL performance model produced a negative contention coefficient as a telltale sign of super-linearity on BigMem clusters.

The subtractive effect of negative σ introduces a point of inflection in the convex superlinear curve that causes it ultimately to become concave, thus crossing over the linear bound at p_c in equation 3. At that point, Hadoop TeraSort superlinear scalability returns to being sublinear in the payback region. The cluster size p_c provides an estimate of the minimal node capacity needed to ameliorate superlinear speedup on BigMem clusters.

Although superlinearity is a bona fide phenomenon, just like perpetual motion it is ultimately a performance illusion. For TeraSort on BigMem, the apparent capacity boost can be traced to successively relaxing the latent IO bandwidth constraint per node as the cluster size grows. This IO bottleneck induces stochastic failures of the HDFS pipeline in the Reduce task. That causes the Hadoop framework to restart the Reduce task file-write, which stretches the measured run-times. If runtime stretching is greatest for T_1 , then successive speedup measurements will be superlinear. Increasing the IO bandwidth per node, as we did with BigDisk clusters, diminishes or eliminates superlinear speedup by reducing T_1 stretching.

This USL analysis suggests super-linear scalability is not peculiar to TeraSort on Hadoop but may arise with any MapReduce application. Superlinear speedup has also been observed in relational database systems.^{2,12} For high-performance computing applications, however, superlinear speedup may arise differently from the explanation presented here.^{4,14,20}

Superlinearity aside, the more important takeaway for many readers may be the following. Unlike most software-

engineering projects, Hadoop applications require only a fixed development effort. Once an application is demonstrated to work on a small cluster, the Hadoop framework facilitates scaling it out to an arbitrarily large number of nodes with no additional effort. For many MapReduce applications, scale-out may be driven more by the need for disk storage than compute power as the growth in data volume necessitates more Maps. The unfortunate term *flat scalability* has been used to describe this effect.²⁵

Although flat scalability may be a reasonable assumption for the initial development process, it does not guarantee that performance goals—such as batch windows, traffic capacity, or service-level objectives—will be met without significant additional effort. The unstated assumption behind the flat-scalability precept is that Hadoop applications scale linearly (Figure 2a) or near-linearly (Figure 2b). Any shuffle-exchange processing, however, will induce a peak somewhere in the scalability profile (Figure 2d). The Hadoop cluster size at which the peak occurs can be predicted by applying the USL to small-cluster measurements. The performance-engineering effort needed to temper that peak will typically far exceed the flat-scalability assumption. As this article has endeavored to show, the USL provides a valuable tool for the software engineer to analyze Hadoop scalability.

Acknowledgments

We thank Comcast Corporation for supporting the acquisition of Hadoop data used in this work. C

Related articles on queue.acm.org

Hazy: Making it Easier to Build and Maintain Big-Data Analytics

Arun Kumar, Feng Niu, and Christopher Ré
<http://queue.acm.org/detail.cfm?id=2431055>

Data-Parallel Computing

Chas. Boyd
<http://queue.acm.org/detail.cfm?id=1365499>

Condos and Clouds

Pat Helland
<http://queue.acm.org/detail.cfm?id=2398392>

References

1. Apache Whirr; <https://whirr.apache.org>.
2. Calvert, C. and Kulkarni D. *Essential LINQ*. Pearson Education, Boston, MA, 2009.

3. Cloudera Hadoop; <http://www.cloudera.com/content/cloudera/en/downloads/cdh/cdh-4-7-0.html/>.
4. Eijkhout, V. *Introduction to High Performance Scientific Computing*. Lulu.com, 2014.
5. Feynman, R.P. Papp perpetual motion engine; <http://hoaxes.org/comments/papparticle2.html>.
6. Gunther, N.J. A simple capacity model of massively parallel transaction systems. In *Proceedings of International Computer Measurement Group Conference*, (1993).
7. Gunther, N.J. A general theory of computational scalability based on rational functions, 2008; <http://arxiv.org/abs/0808.1431>.
8. Gunther, N.J. *Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services*. Springer, New York, NY, 2007.
9. Gunther, N.J. Performance and scalability models for a hypergrowth e-commerce Web site. *Performance Engineering*, R.R. Dumke, C. Rautenstrauch, A. Schmietendorf, and A. Scholz, eds. A Lecture Notes in Computer Science 2047 (2001). Springer-Verlag 267-282.
10. Gunther, N.J. PostgreSQL scalability analysis deconstructed. *The Pith of Performance*, 2012; <http://perfdynamics.blogspot.com/2012/04/postgresql-scalability-analysis.html>.
11. Gunther, N.J., Subramanyam, S. and Parvu, S. Hidden scalability gotchas in memcached and friends. *VELOCITY Web Performance and Operations Conference*, (2010).
12. Haas, R. Scalability, in graphical form, analyzed, 2011; <http://rhaas.blogspot.com/2011/09/scalability-in-graphical-form-analyzed.html>.
13. Hadoop Log Tools; <https://github.com/melrief/Hadoop-Log-Tools>.
14. Hennessy, J.L. and Patterson, D.A. *Computer Architecture: A Quantitative Approach*. Second edition. Morgan Kaufmann, Waltham, MA, 1996.
15. O'Malley, O. TeraByte sort on Apache Hadoop, 2008; <http://sortbenchmark.org/YahooHadoop.pdf>.
16. O'Malley, O., Murthy, A. C. 2009. Winning a 60-second dash with a yellow elephant; <http://sortbenchmark.org/Yahoo2009.pdf>.
17. Performance Dynamics Company. How to quantify scalability, 2014; <http://www.perfdynamics.com/Manifesto/USLscalability.html>.
18. Schwartz, B. Is VoltDB really as scalable as they claim? Percona MySQL Performance Blog; <http://www.percona.com/blog/2011/02/28/is-voltdb-really-as-scalable-as-they-claim/>.
19. sFlow. SDN analytics and control using sFlow standard — Superlinear; <http://blog.sflow.com/2010/09/superlinear.html>.
20. Stackoverflow. Where does superlinear speedup come from?; <http://stackoverflow.com/questions/4332967/where-does-super-linear-speedup-come-from>.
21. Sun Fire X2270 M2 superlinear scaling of Hadoop TeraSort and CloudBurst benchmarks, 2010; https://blogs.oracle.com/BestPerf/entry/20090920_x2270m2_hadoop.
22. Sutter, H. Going superlinear. *Dr. Dobbs's J.* 33, 3 (2008); <http://www.drdoobs.com/cpp/going-superlinear/206100542>.
23. Sutter, H. Super linearity and the bigger machine. *Dr. Dobbs's J.* 33, 4 (2008); <http://www.drdoobs.com/parallel/super-linearity-and-the-bigger-machine/206903306>.
24. White, T. *Hadoop: The Definitive Guide*, third edition. O'Reilly Media, 2012.
25. Yahoo! Hadoop Tutorial; <https://developer.yahoo.com/hadoop/tutorial/module1.html#scalability>.

Neil J. Gunther (<http://perfdynamics.blogspot.com>; tweets as @DrOz) is a researcher and teacher at Performance Dynamics where he developed the USL and the PDQ open source performance analyzer.

Paul Puglia (pjpuglia@gmail.com) has been working in IT for more than 20 years doing Python programming, system administration, and performance testing. He has authored an R package, SATK, for fitting performance data to the USL, and contributed to the PDQ open source performance analyzer.

Kristofer Tomasette (ktomasette@gmail.com) is a senior software engineer on the Platforms & APIs team at Comcast Corporation. He has built software systems involving warehouse management, online banking, telecom, and most recently cable TV.

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

Speaking military jargon, users can create labels and draw symbols to position objects on digitized maps.

**BY PHILIP R. COHEN, EDWARD C. KAISER,
M. CECILIA BUCHANAN, SCOTT LIND,
MICHAEL J. CORRIGAN, AND R. MATTHEWS WESSON**

Sketch-Thru-Plan: A Multimodal Interface for Command and Control

IN 2000, OVIATT and Cohen²⁵ predicted multimodal user interfaces would “supplement, and eventually replace, the standard GUIs of today’s computers for many applications,” focusing on mobile interfaces with alternative modes of input, including speech, touch, and handwriting, as well as map-based interfaces designed to process and fuse multiple simultaneous modes. In the intervening years, basic multimodal interfaces employing alternative input modalities have indeed become the dominant interface for mobile devices. Here, we describe an advanced fusion-based multimodal map system called Sketch-Thru-Plan, or STP, developed from 2009 to 2011 under the DARPA Deep Green program, enabling rapid creation of operational plans during command and control (C2) for military ground operations. As background, we describe

the challenges posed by ground operations for C2 systems and their user interfaces. We discuss how C2 GUIs have led to inefficient operation and high training costs. And to address them, we cover STP’s multimodal interface and evaluations. Finally, we discuss deployment of the system by the U.S. Army and U.S. Marine Corps. This case study involves the user-centered design-and-development process required for promising basic research to scale reliably and be incorporated into mission-critical products in large organizations.

Command and Control

Command-and-control software must meet the needs of the commander and many types of staff, ranging from higher-echelon commanders (such as of an Army division or brigade) and their own dedicated staff to relatively inexperienced commanders of smaller units.¹ Across this range, there is great need for a planning tool that is easy to learn and use for both actual and simulated operations while being functional in field and mobile settings with varying digital infrastructure and computing devices. No military C2 system currently meets all these requirements, due in part to GUI limitations.

Prior to the introduction of digital systems, C2 functions were performed on paper maps with transparent plastic overlays and grease pencils. Users would collaboratively develop plans by speaking to one another while drawing on a map overlay. Such an interface had the benefit of requiring no interface training and fail-safe operation. How-

» key insights

- **Multimodal interfaces allow users to concentrate on the task at hand, not on the tool.**
- **Multimodal speech+sketch interfaces employing standardized symbol names and shapes can be a much more efficient means of creating and positioning symbols on maps during planning.**
- **Nearly all users tested preferred the multimodal interface to the graphical user interfaces commonly used for command-and-control and planning.**



STP

2 March 2015
9.33

• 579,1

ever, obvious drawbacks included the need to copy data into digital systems and lack of remote collaboration. Addressing them, C2 systems today are based on GUI technologies. The most widely used Army C2 system, called the Command Post of the Future, or CPOF,¹¹ is a three-screen system that relies on a drag-and-drop method of manipulating information. It supports co-located and remote collaboration through human-to-human dialogue and collaborative sketching. CPOF was a major advance over prior C2 systems^a and the primary Army C2 system during Operation Iraqi Freedom, starting 2003.

Table 1 outlines how a CPOF user would send a unit to patrol a specified route from time 0800 to 1600. These

a http://www.army.mil/article/16774/Command_Post_of_the_Future_Wins_Outstanding_US_Government_Program_Award/

11 high-level steps can take an experienced user one minute to perform, with many more functions necessary to properly specify a plan. In comparison, with a version of the Quickset multimodal interface that was tightly integrated with CPOF in 2006, a user could say “Charlie company, patrol this route <draw route> from oh eight hundred to sixteen hundred.” All attribute values are filled in with one simple utterance processed in six seconds on a tablet PC computer.

Soldiers must learn where the many functions are located within the menu system, how to link information by “ctrl-dragging”^b a rendering of it,

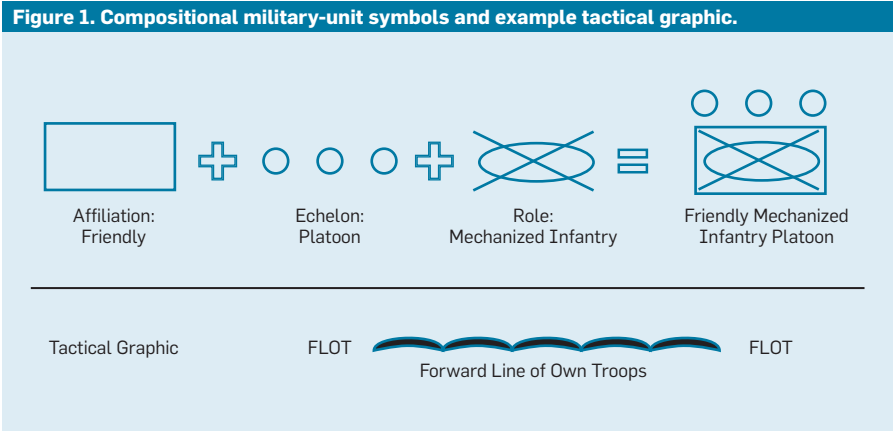
b “Ctrl-dragging” refers to holding down the CTRL key while also holding the left mouse button on a map symbol, then dragging the symbol elsewhere in the user interface; a “clone” of the symbol appears at the destination location, such that if the original one is changed, the clone is changed as well.

and how to navigate among various screens and windows. With CPOF requiring so many atomic GUI steps to accomplish a standard function, SRI International and General Dynamics Corp. built a learning-by-demonstration system an experienced user could use to describe a higher-level procedure.²¹ Expert users were trained within their Army units to create such procedures, and the existence of the procedures would be communicated to the rest of the unit as part of the “lore” of operating the system. However, if the interface had supported easier expression of user intent, there would have been less need for the system to learn higher-level procedures. Thousands of soldiers are trained at great expense in Army schoolhouses and in deployed locations each year to operate this complex system.

One essential C2 planning task is to position resources, as represented by symbols on a map of the terrain. Symbols are used to represent military units, individual pieces of equipment, routes, tactical boundaries, events, and tasks. The symbol names and shapes are part of military “doctrine,” or standardized procedures, symbols, and language, enabling people to share meaning relatively unambiguously. Soldiers spend considerable time learning doctrine, and anything that reinforces doctrine is viewed as highly beneficial.

Each unit symbol has a frame and color (indicating friendly, hostile, neutral, and coalition), an echelon marking (such as a platoon) on top, a label or “designator” on the side(s), and a “role” (such as armored, medical, engineering, and fixed-wing aviation) in the middle, as well as numerous other markings (see Figure 1). This is a compositional language through which one can generate thousands of symbol configurations. In order to cope with the large vocabulary using GUI technology, C2 systems often use large dropdown menus for the types of entities that can be positioned on the map. Common symbols may be arrayed on a palette a user can select from. However, these palettes can become quite large, taking up valuable screen space better used for displaying maps, plans, and schedules.

Table 1. Steps to send a unit on patrol through Command Post of the Future.	
Step	CPOF
1.	Right click on the display background (not the map) to bring up a menu of CPOF objects that can be created;
2.	Select “Task,” opening a small window;
3.	Type the task label “Patrol”;
4.	Type the name of the unit in the “Performer” slot (such as C/1-62/3-BCT);
5.	Use the mouse to sketch the route on a map in digital ink;
6.	Holding down the control key, select the digital ink just drawn (with left mouse button) and drag it into the container box in the task window;
7.	Cntrl-drag the task window to the map near the link to position a symbol for the Task on the map;
8.	Locate and expose the “Schedule” window;
9.	Cntrl-drag the task to the Schedule window, positioning the Task in the line associated with the name of the unit in the Performer slot;
10.	Move the task graphic to the day in question; and
11.	Slide the left and right interval boundaries to line up with 0800 and 1600, respectively.



Another method used in GUIs to identify a military unit involves specifying its compositional pieces in terms of the attributes and values for unit name, role, echelon, and strength. Each is displayed with multiple smaller menus from which the user chooses a value. The user may type into a search field that finds possible units through a string match. The user must still select the desired entity and set any attribute values via menus. When a symbol is created or found, it is then positioned through a drag-and-drop operation onto the map. Due to these constraints (and many more) on system design, users told STP developers C2 system interfaces based on such classical GUI technologies are difficult to learn and use. We have found speech-and-sketch interfaces employing doctrinal language, or standardized symbol names and shapes, to be a much more efficient means for creating and positioning symbols.

Multimodal Map-Based Systems

Many projects have investigated multimodal map-based interaction with pen and voice^{3,5,6,14,20,23} and with gesture and voice.^{2,7,16,19} Some such systems represent the research foundation for the present work, though none to our knowledge is deployed for C2. Apart from smartphones, the most widely deployed multimodal system is Microsoft's Kinect, which tracks the user's body movements and allows voice commands, primarily for gaming and entertainment applica-

tions; enterprise and health applications are also beginning to appear;²² and other commercial multimodal systems have been developed for warehousing and are emerging in automobiles.²⁶ Adapx's STP work is most related to the QuickSet system developed at the Oregon Graduate Institute in the late 1990s.^c QuickSet^{5,6,14,23-25} was a prototype multimodal speech/sketch/handwriting interface used for map-based interaction. Because speech processing needs no screen space, its multimodal interface was easily deployed on tablets, PDAs, and wearables, as well as on wall-size displays. Offering distributed, collaborative operation, it was used to position entities on a map by speaking and/or drawing, as well as create tasks for them that could be simulated through the Modular Semi-Automated Forces, or ModSAF,^{4,8} simulator. QuickSet was also used to control 3D visualizations,⁷ various networked devices (such as TV monitors and augmented-reality systems¹⁶) through hand gestures tracked with acoustic, magnetic, and camera-based methods.

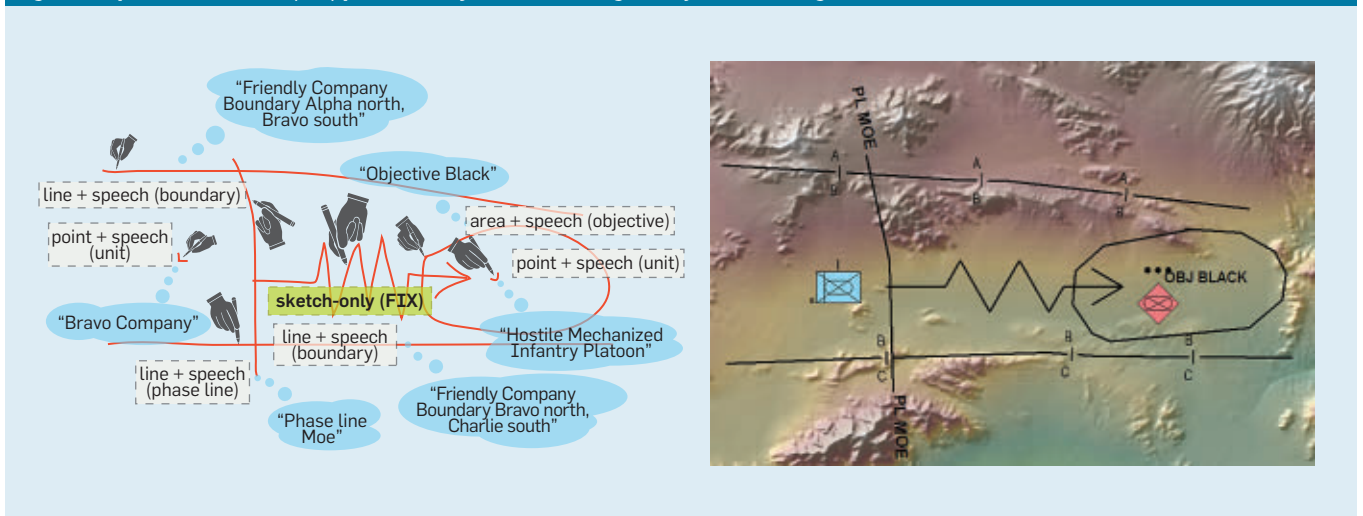
Based on extensive user-centered-design research, the Oregon Graduate Institute team showed users prefer to interact multimodally when manipulating a map. They are also able to select the best mode or combination of modes to suit their situation

and task.^{23,25} User sketching typically provides spatial information (such as shape and location), while speech provides information about identity and other attributes. This user interface emulates the military's non-digital practices using paper maps⁶ and leads to reduced cognitive load for the user.²³

QuickSet's total vocabulary was approximately 250 unit symbols and approximately 180 "tactical graphics," as in Figure 1. Speech recognition was based on an early IBM recognizer, and sketch recognition involved a lightly trained neural network and hidden Markov-model recognizer. The major research effort was devoted to establishing innovative methods for multimodal fusion processing.¹⁴ QuickSet's unification-based fusion of multimodal inputs¹⁴ supported mutual disambiguation, or MD, of modalities^{16,24} in which processing of information conveyed in one mode compensated for errors and ambiguities in others, leading to relative error rate reduction of 15%–67%; for example, a sketch with three objects could disambiguate that the user said "boats" and not "boat." MD increased system robustness to recognition errors, critical in high-noise environments, where users are heavily accented, or when sketches are created while moving or when the user's arm is tired.¹⁸ QuickSet demonstrated a multimodal interface could function robustly under such real-world conditions, a necessary precondition of field deployment.

c Adapx Inc. was a corporate spin-off of the Oregon Graduate Institute's parent institution, the Oregon Health and Science University, Portland, OR.

Figure 2. Speech and Sketch (left) processed by STP into the digital objects on the right.



Sketch-Thru-Plan

DARPA established the Deep Green program in 2008 with the aim of using simulation during mission planning to enable planners to play out the consequences of a course of action, or COA, against the most likely and most dangerous enemy COAs. A COA is a specification of the actions a set of entities will perform over time to accomplish a mission. A critical piece of the Deep Green effort was to develop an easy-to-use interface that would allow a planning team to create COAs rapidly. DARPA chose Adapx's multimodal technology, a derivative of QuickSet, for this task, along with the companies SAIC (<http://www.saic.com>) and BAE Systems (<http://www.baesystems.com>). Finally, as prime contractor, Adapx developed the STP system with guidance from a team of subject-matter experts and testing by ROTC students. STP users develop their plans by speaking and drawing, with an optional GUI input if appropriate. STP inter-operates with existing C2 systems, notably CPOF, the LG-RAID simulator,²⁷ and visualizations based on Google Earth, populating them with planned entity positions and tasks.

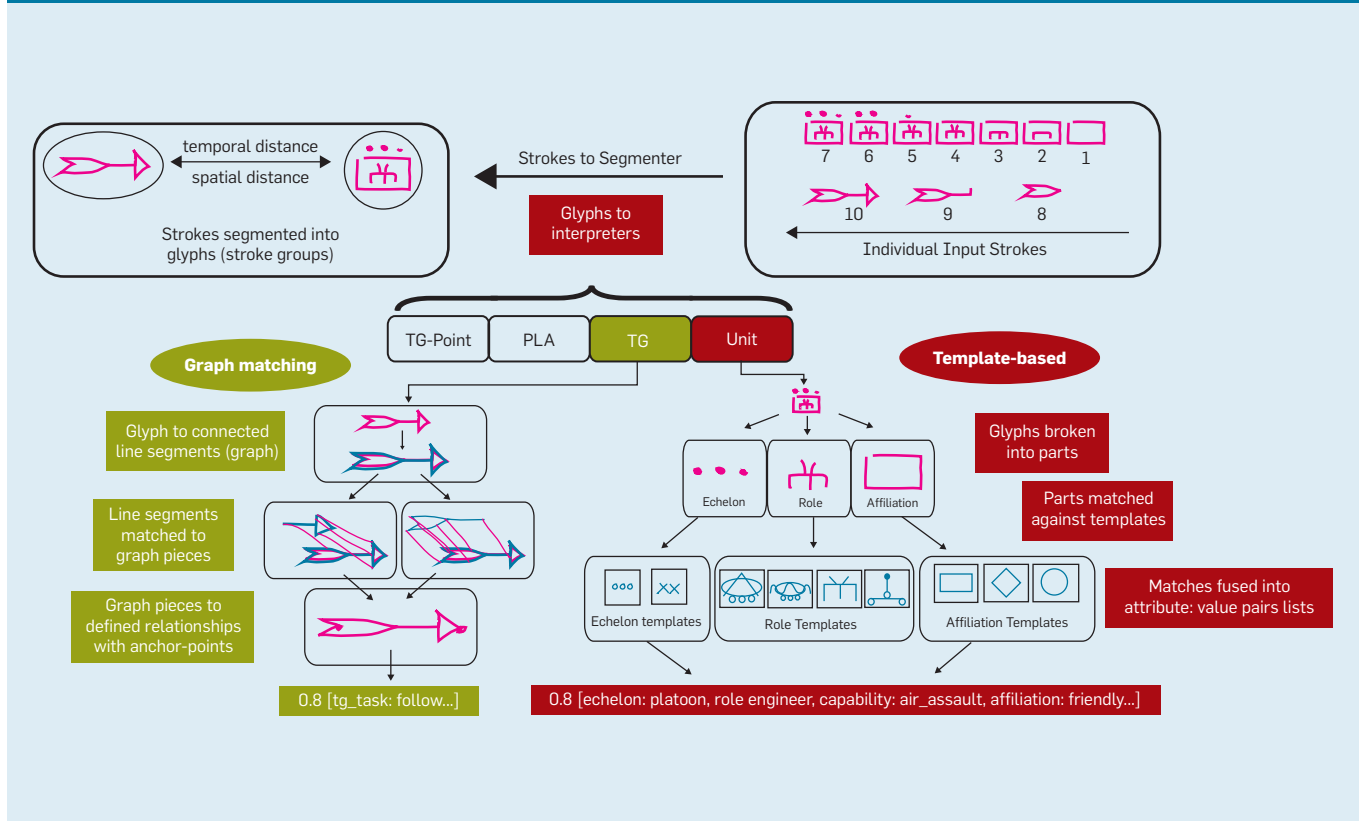
Sketched inputs are captured through stylus, finger, mouse, or digital pen and paper inputs. Most people prefer both speech and sketch, but sketch alone can be very useful, especially with digital paper and pen. The fastest and easiest input style is to sketch a simple point, line, or area where a symbol should be placed while saying the symbol's features (such as status, affiliation, role, strength, and designation); for example, in Figure 2 (left panel), the user can speak the type of unit or tactical graphic—"hostile mechanized infantry platoon," "objective black," "company boundary alpha north bravo south"—while providing a point, line, or area drawing. The user can also draw the symbols, as in the "Fix" symbol, or zigzag arrow in Figure 2. These multimodal inputs are recognized and fused to provide military symbols, as in Figure 2, right panel. The resulting symbols are not just icons on a bitmap but digitally geo-registered objects that are part of the system's databases and populate C2 and simulation systems.

The STP multimodal interface recognizes more than 4,000 symbol and

tactical graphic configurations, as in Figure 1, each with a set of attributes and values. It also recognizes more than 150 tactical tasks (such as patrolling a route and delivering supplies) that make use of spatially and semantically related groups of symbols. The symbols are provided to both the speech and sketch recognizers through a database that specifies related labels, icons, and unique identifiers. The recognizer vocabularies are populated automatically, enabling the system to adopt additional symbols or switch to new domains.

STP enables a team of users to develop a plan collaboratively, with different users serving various functional roles (such as commander, logistics, intelligence, and engineering) and contribute their portions of the overall plan. The system supports users in creating notional entities (such as a default rifle company) or positioning actual entities from an existing military organization, assigning and synchronizing tasks, filling out worksheets that populate role-specific systems (such as for logistics and engineering), and creating required documents (such as an operations order).

Figure 3. Overview of sketch-recognition processing: TG = tactical graphic, PLA = point-line-area.



The scope of STP's vocabulary and task coverage is an order of magnitude greater than that of QuickSet. It also uses more capable speech recognition, sketch recognition, and mapping subsystems.^d Like QuickSet, STP supports the same multimodal interface across multiple form factors, including handheld and tablet computers, as well as digital pen and paper.^e STP's design is informed by domain experts, resulting in a full planning tool. Rather than being a research prototype, it has been developed to a higher level of technology readiness,^e and STP has been tested by actual operational military units. As a result, STP is being transitioned to multiple organizations within the U.S. military.

STP Components

The system's major functional components are covered in the following sections.

Speech and natural language. The goal of STP's spoken-language processing is support for multimodal language, which is briefer and less complex than speech-only constructions.²⁵ The aim is to provide significantly improved user performance, be transparent and memorable to the user, and minimize cognitive load and recognition errors. The STP approach is as follows: Users typically speak noun phrases and draw symbols to create and position entities. The basic spoken language vocabulary of nouns with their attributes and values is defined in the database, as discussed earlier.

When creation of an entity involves multiple attributes, users are encouraged to not impart all information in one long utterance, as it can lead to syntactic complexity, "disfluencies," and other obstacles to robust spoken interaction. Instead, users can select an entity by drawing a mark on it and speaking subsequent attribute-value



The STP multimodal interface recognizes more than 4,000 symbol and tactical graphic configurations, each with a set of attributes and values.



information^f; for example, a unit's strength can be altered by marking it and saying "strength reduced." Likewise, a user can create a restricted operating zone for aviation for a certain time period and altitude with "R O Z <draw area> from oh eight hundred to sixteen hundred," <mark the ROZ> "minimum altitude one thousand meters maximum two thousand meters." For most C2 systems, creating this 3D tactical graphic is very time consuming.

Unlike QuickSet, STP includes two natural language parsers, one handling just noun phrases, whose constituent phrases can occur in any order, and Gemini,⁹ which uses a large-scale grammar of English. The noun phrase parser is used during symbol creation for analyzing such phrases as "anticipated enemy mechanized platoon strength reduced." The broader-coverage Gemini parser is used when describing tasks for the entities that typically involve uttering verb phrases (such as "Resupply along MSR alpha"). Gemini has been used in many systems, including at NASA,¹⁰ and is one of the most competent parsers of English designed for spoken-language systems. The verb phrases in the grammar are derived from a "task signature" table that specifies the types of required and optional arguments for each military task. Because the system can infer potential tasks, the need for a user to utter complex sentences is minimized.

Speech recognition employs the Microsoft Speech engine in Windows 7/8 using grammar-based recognition. The system is always listening, but speech recognition is coordinated with screen touching, so human-to-human conversation without sketch does not result in spurious recognition. STP coordinates multiple simultaneous instances of the recognizer, each with a different grammar, or "context," as a function of the user

^d Unlike QuickSet's use of a geo-registered bit-map, STP uses the ArcGIS mapping system from ESRI.

^e For NASA's definition of "technology readiness level," see http://esto.nasa.gov/files/trl_definitions.pdf. QuickSet was developed through technology readiness level 3, whereas STP has been developed through level 6. Ongoing development and deployment will take it through take it to level 9.

^f Note "selection" through marking is not an atomic operation but must be recognized and interpreted, as a stroke can be ambiguous. Selection through marking avoids having a "moded" interface in which selection is a distinct mode; likewise, STP does not use hardware-assisted selection (such as a specific button on a pen or keyboard) to support simple touch or digital pen and paper.

interface state. Contextual knowledge restricts potential speech and language, thus increasing accuracy and speed; for example, an “attribute-value” grammar context is invoked when a stroke is drawn over an object on the map. As the context-setting actions may themselves be ambiguous, STP is designed to compare the results of multiple simultaneous recognizers embodying different restrictions.

In the future, it may be helpful to use spoken dictation, as in Google Voice, Nuance Communications’s Dragon Dictate, Apple’s Siri, and speech-to-speech translation systems,¹³ that require development of large-scale statistical-language models. However, because the spoken military data needed to build such language models is likely classified, this approach to creating a language model could be problematic. Since STP can take advantage of users’ knowledge of military jargon and a structured planning process,⁸ grammar-based speech recognition has thus far been successful.

Sketch recognition. STP’s sketch recognizer is based on algorithms from computer vision, namely Hausdorff matching,¹⁷ using an array of ink interpreters to process sketched symbols and tactical graphics (see Figure 3). For unit symbols, the recognizer’s algorithm uses templates of line segments, matching the sketched digital ink against them and applying a modified Hausdorff metric based on stroke distance and stroke angles to compute similarity. For tactical graphics, the recognizer creates graphs of symbol pieces and matches them against the input. Fundamental to them all is a spatiotemporal ink segmenter. Regarding spatial segmentation, if the minimum distance of a given stroke from the currently segmented group of strokes, or “glyph,” is below a threshold proportional to the already existing glyph size and its start time is within a user-settable threshold from the end-time of the prior stroke, then the new stroke is added to the existing glyph.

For template-based unit-icon interpretation, the affiliation “frame”



The interface takes advantage of and reinforces skills soldiers already have, as they are trained in the standardized language, symbols, and military decision-making process.



is first located, after which the glyph is broken into its constituent parts, including affiliation, role, and echelon, that have canonical locations relative to the frame. Though the roles may themselves have compositional structure, they are matched holistically. Where linguistic content that annotates the icon is conventionally expected, handwriting is processed by Microsoft’s recognizer. These parts are then compared to a library of template images, with the results combined to form recognition outputs. If a symbol “frame” is not found, the sketch recognizer attempts to use the tactical graphics interpreter. For tactical graphics, whose shapes can be elongated or contorted, the algorithm uses a graph-matching approach that first partitions the glyph into a graph of line segments and nodes. This graph is then matched against piecewise graph templates that allow for elongation or bending. The pieces are recombined based on sketch rules that define the relations between the pieces and anchor points from which a complete symbol can be composed; for example, such rules define a “forward line of own troops,” or FLOT, symbol, as in Figure 1, as a “linear array” of semicircles (a “primitive”), with a barbed-wire fence composed of two approximately parallel lines, plus a parallel linear array of circles.

An advantage of the template-based approach to unit-icon recognition is easy expansion by adding new templates to the library; for example, new unit roles can be added in the form of scalable vector graphics that would then be located within the affiliation border by the compositional unit symbol recognizer.

Explicit and implicit task creation.

Aside from creating and positioning symbols on a map, users can state tasks explicitly or rely on the system to implicitly build up an incremental interpretation of the set of tasks that use those symbols (see Figure 4). STP does the latter inference by matching the symbols on the map against the argument types of possible domain tasks (such as combat service units perform “supply” and medical units perform “evacuate casualties”), as in Figure 4, subject to spatiotemporal

⁸ Soldiers are taught to use the structured Military Decision-Making Process.²⁸

constraints. STP presents the planner with a running visualization of matching tasks in the evolving plan under creation. The planner can readily inspect the potential tasks, accepting or correcting them as needed. Here, STP has inferred that the Combat Service Support unit and Main Supply Route A can be combined into the task Resupply along Main Supply Route A. If that is correct, the planner can select the checkbox that then updates the task matrix and schedule. As the planner adds more symbols to the map, the system's interpretations of matching tasks are likewise updated. Task start and end times can be spoken or adjusted graphically in a standard Gantt chart task-synchronization matrix. Note STP is not trying to do automatic planning or plan recognition but rather assist during the planning process; for instance, STP can generate a templated "operations order" from the tasks and graphics, a required output of the planning process. Much more planning assistance can, in principle, be provided, though not clear is what a planner would prefer.

Because the system is database driven, the multimodal interface and system technology have many potential commercial uses, including other types of operations planning (such as "wildland" firefighting, as firefighters say), as well as geographical information management, computer-aided design, and construction management.

Evaluations

Four types of evaluations of STP have been conducted by the U.S. military: component evaluations, user juries, controlled study, and exercise planning tests.

Component evaluations. In recognition tests of 172 symbols by a DARPA-selected third-party evaluator during the Deep Green Program in 2008, the STP sketch-recognition algorithm had an accuracy of 73% for recognizing the correct value at the top of the list of potential symbol-recognition hypotheses. The next-best Deep Green sketch recognizer built for the same symbols and tested at the same time with the same data had a 57% recognition accuracy for the top-scoring hypothesis.¹² Rather than

use sketch alone, most users prefer to interact multimodally, speaking labels while drawing a point, line, or area. STP's multimodal recognition in 2008, as reported by an externally contracted evaluator, was a considerably higher 96%. If STP's interpretation is incorrect, users are generally able to re-enter the multimodal input, select among symbols on a list of alternative symbol hypotheses, or invoke the multimodal help system that presents the system's coverage and can be used for symbol creation.

STP has also been tested using head-mounted noise-canceling microphones in high-noise Army vehicles. Two users—one male, one female—issued a combined total of 221 multimodal commands while riding in each of two types of moving vehicles in the field, with mean noise 76.2dba and spikes to 93.3dba. They issued the same 220 multimodal commands to STP with the recorded vehicle noise played at maximum volume in the laboratory, with mean 91.4dba and spikes to 104.2dba. These tests

Figure 4. Implicit task creation.

The map depicts a combat-service-support unit and main supply route, medical unit, and casualty-collection point; the task inference process finds tasks (such as resupply) that can include various entities along main supply route A.

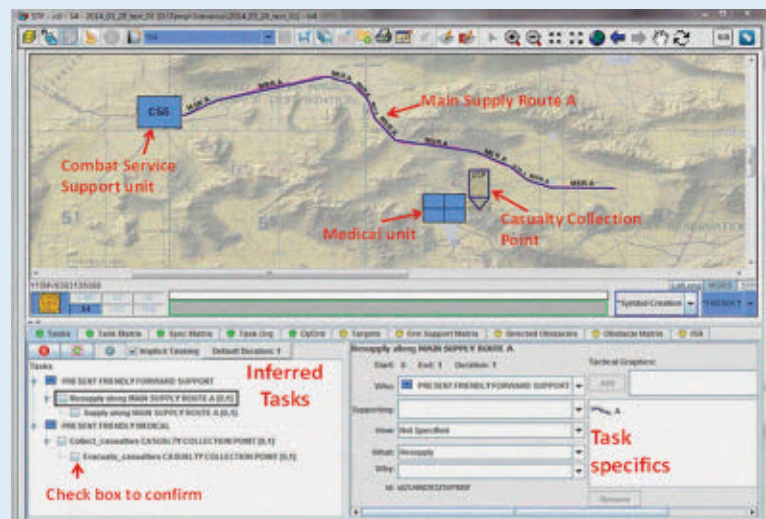
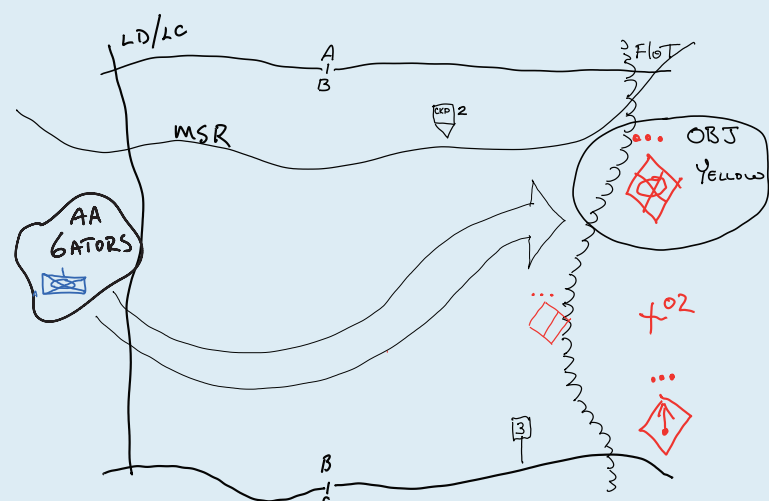


Figure 5. COA sketch used in controlled study, January 2013.



resulted in 94.5% and 93.3% multimodal recognition accuracy, respectively. We conjecture that, in addition to the multimodal architecture, the noise-canceling microphones may have compensated for the loud but relatively constant vehicle noise. Further research by the STP team will look to tease apart the contributions of these factors in a larger study.

User juries. One way the Army tests software is to have soldiers just returning from overseas deployment engage in a “user jury” to try a potential product and provide opinions as to whether it would have been useful in their recent activities. In order to get soldier feedback on STP, 2011–2013, the Army’s Training and Doctrine Command invited 126 soldiers from four Army divisions experienced with the vehicle C2 system and/or CPOF to compare them with STP. For privacy, this article has changed the names of those divisions to simply Divisions 1, 2, 3, and 4. STP developers trained soldiers for 30 minutes on STP, then gave them a COA sketch to enter using STP. They later filled out a five-point Likert-style questionnaire. In all areas, STP was judged more usable and preferred to the soldiers’ prior C2 systems; Table 2 summarizes their comparative ratings of STP versus their prior C2 systems.

Controlled user study. Contractors have difficulty running controlled studies with active-duty soldiers. However, during the STP user jury in January 2013, 12 experienced CPOF users from Division 3 evaluated STP vs. CPOF in a controlled experiment. Using a within-subject design with order-of-system-use counterbalanced, experienced CPOF users were

trained for 30 minutes by the STP development team on STP, then given the COA sketch in Figure 5 to enter through both STP and CPOF. Results showed these experienced CPOF users created and positioned units and tactical graphics on the map using STP’s multimodal interface 235% faster than with CPOF^h; these subjects’ questionnaire remarks are included in Table 2. Note “symbol laydown” is only one step in the planning process, which also included tasking of units and creating a full COA and an operations order. Experts have reported the STP time savings for these other planning functions is considerably greater.

Exercise planning test. STP was recently used by a team of expert planners charged with developing exercise COAs that would ultimately appear in CPOF. The STP team worked alongside another expert planning team using Microsoft PowerPoint in preference to CPOF to develop its plans. Many attempts to develop exercise COAs have used various planning tools, including CPOF itself, but PowerPoint continues to be used in spite of its many limitations (such as lack of geospatial fidelity) because it is known by all. When the exercise was over, the team using PowerPoint asked for STP for planning future exercises.

Transition and Deployment

Although the U.S. military is extremely conservative in its adoption of computing technology, there is today a growing appreciation that operational efficiency and training are

being hampered by systems with different user interfaces and operational difficulty. Still, such a realization takes time to pervade such a large organization with so many military and civilian stakeholders, including operational users and the defense-acquisition community. In addition to technology development, it took the STP development team years of presentations, demonstrations, tests, and related activities to achieve the visibility needed to begin to influence organizational adoption. Over that time, although the STP prototypes had been demonstrated, commercial availability of speech recognition was necessary to enable conservative decision makers to decide that the risk from incorporating speech technology into mission-critical systems had been sufficiently reduced. Moreover, the decision makers had independently become aware of the effects of interface complexity on their organizations’ training and operations. Still, the process is by no means complete, with organizational changes and thus customer education always a potential problem. Currently, STP has been transitioned to the Army’s Intelligence Experimentation Analysis Element, the Army Simulation and Training Technology Center, and the Marine Corps Warfighting Laboratory’s Experiments Division where it is used for creating plans for exercises and integrating with simulators. We have also seen considerable interest from the Army’s training facilities, where too much time is spent training students to use C2 systems, in relation to time spent on the subject matter. Moreover, beyond STP’s use as a planning tool, there has been great interest in its multimodal technology for rapid data entry, in both vehicle-based computers and handhelds.

Regarding full deployment of STP, the congressionally mandated “program of record” acquisition process specifies program budgets many years into the future; new technologies have a difficult time being incorporated into such programs, as they must become an officially required capability and selected to displace already budgeted items in a competitive feature triage process. In spite of these hurdles, STP and multimodal

^h F-test two-sample for variances test: $F(19) = 4.05$, $p < 0.02$

Table 2. Questionnaire results for STP vs. prior C² system(s) used by the subjects.

Organization	System Compared to STP	Number of Users	STP Easier to Use	STP Faster	STP Better	Prefer Speech/Sketch
Division 1	Vehicle C2 system	41	83%	88%	81%	90%
Division 2	Vehicle C2 system	44	97%	97%	100%	87%
Division 3	Vehicle C2 system	37	78%	89%	84%	87%
	Overall	122	87%	92%	89%	88%
Division 2	CPOF	16	76%	94%	85%	88%
Division 3	CPOF	12	88%	79%	84%	100%
Division 4	CPOF	5	100%	100%	100%	100%
	Overall	33	84%	89%	87%	94%

interface technology are now being evaluated for integration into C2 systems by the Army's Program Executive Office responsible for command-and-control technologies.

Conclusion

We have shown how the STP multimodal interface can address user-interface problems challenging current C2 GUIs. STP is quick and easy to learn and use and supports many different form factors, including handheld, tablet, vehicle-based, workstation, and ultra-mobile digital paper and pen. The interface takes advantage of and reinforces skills soldiers already have, as they are trained in the standardized language, symbols, and military decision-making process. In virtue of this common "doctrinal" language, STP users can quickly create a course of action or enter data multimodally for operations, C2, and simulation systems without extensive training on a complicated user interface. The result is a highly usable interface that can be integrated with existing C2 systems, thus increasing user efficiency while decreasing cost.

Acknowledgments

STP development was supported by Small Business Innovation Research Phase III contracts, including HR0011-11-C-0152 from DARPA, a subcontract from SAIC under prime contract W15P7T-08-C-M011, a subcontract from BAE Systems under prime contract W15P7T-08-C-M002, and contract W91CRB-10-C-0210 from the Army Research, Development, and Engineering Command/Simulation and Training Technology Center. This article is approved for public release, distribution unlimited. The results of this research and the opinions expressed herein are those of the authors, and not those of the U.S. Government. We thank Todd Hughes, Colonels (ret.) Joseph Moore, Pete Corpac, and James Zanol and the ROTC student testers. We are grateful to Paulo Barthelme, Sumithra Bhakthavatsalam, John Dowding, Arden Gudger, David McGee, Moiz Nizamuddin, Michael Robin, Melissa Trapp-Petty, and Jack Wozniak for their contributions to developing and testing of STP. Thanks also to Sharon

Oviatt, General (ret.) Peter Chiarelli, and the anonymous reviewers. 

References

1. Alberts, D.S. and Hayes, R.E. *Understanding Command and Control*. DoD Command and Control Research Program Publication Series, Washington, D.C., 2006.
2. Bolt, R.A. Voice and gesture at the graphics interface. *ACM Computer Graphics* 14, 3 (1980), 262–270.
3. Cheyer, A. and Julia, L. Multimodal maps: An agent-based approach. In *Proceedings of the International Conference on Cooperative Multimodal Communication* (Eindhoven, the Netherlands, May). Springer, 1995, 103–113.
4. Clarkson, J.D. and Yi, J. LeatherNet: A synthetic forces tactical training system for the USMC commander. In *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation Technical Report IST-TR-96-18*, University of Central Florida, Institute for Simulation and Training, Orlando, FL, 1996, 275–281.
5. Cohen, P.R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Chen, L., and Clow, J. QuickSet: Multimodal interaction for distributed applications. In *Proceedings of the Fifth ACM International Conference on Multimedia* (Seattle, WA, Nov. 9–13). ACM Press, New York, 1997, 31–40.
6. Cohen, P.R. and McGee, D.R. Tangible multimodal interfaces for safety-critical applications. *Commun. ACM* 47, 1 (Jan. 2004), 41–46.
7. Cohen, P.R., McGee, D., Oviatt, S., Wu, L., Clow, J., King, R., Julier, S., and Rosenblum, L. Multimodal interaction for 2D and 3D environments. *IEEE Computer Graphics and Applications* 19, 4 (Apr. 1999), 10–13.
8. Courtemanche, A.J. and Czeranowicz, A. ModSaf development status. In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, University of Central Florida, Institute for Simulation and Training, Orlando, FL, 1995, 3–13.
9. Dowding, J., Gawron, J.M., Appelt, D., Bear, J., Cherny, L., Moore, R., and Moran, D. Gemini: A natural language system for spoken-language understanding. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics* (Ohio State University, Columbus, OH, June 22–26). Association for Computational Linguistics, Stroudsburg, PA, 1993, 54–61.
10. Dowding, J., Frank, J., Hocky, B.A., Jonsson, A., Aist, G., and Hieronymus, J. A spoken-dialogue interface to the EUROPA planner. In *Proceedings of the Third International NASA Workshop on Planning and Scheduling for Space* (Washington, D.C.). NASA, 2002.
11. Greene, H., Stotts, L., Patterson, R., and Greenburg, J. *Command Post of the Future: Successful Transition of a Science and Technology Initiative to a Program of Record*. Defense Acquisition University, Fort Belvoir, VA, Jan. 2010; <http://www.dau.mil>
12. Hammond, T., Logsdon, D., Peschel, J., Johnston, J., Taele, P., Wolin, A., and Paulson, B. A sketch-recognition interface that recognizes hundreds of shapes in course-of-action diagrams. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems* (Atlanta, Apr. 10–15). ACM Press, New York, 2010, 4213–4218.
13. Hyman, P. Speech-to-speech translations stutter, but researchers see mellifluous future. *Commun. ACM* 57, 4 (Apr. 2014), 16–19.
14. Johnston, M., Cohen, P.R., McGee, D., Oviatt, S.L., Pittman, J.A., and Smith, I. Unification-based multimodal integration. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Annual Meeting of the European ACL* (Madrid, Spain, July 7–12). Association for Computational Linguistics, Stroudsburg, PA, 1997, 281–288.
15. Johnston, M., Bangalore, S., Varireddy, G., Stent, A., Ehlen, P., Walker, M., Whittaker, S., and Maloor, P. MATCH: An architecture for multimodal dialogue systems. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (Philadelphia, PA, July). Association for Computational Linguistics, Stroudsburg, PA, 2002, 376–383.
16. Kaiser, E., Olwal, A., McGee, D., Benko, H., Corradini, A., Li, X., Cohen, P.R., and Feiner, S. Mutual disambiguation of 3D multimodal interaction in augmented and virtual reality. In *Proceedings of the Seventh International Conference on Multimodal Interfaces* (Trento, Italy, Oct. 4–6). ACM Press, New York, 2005, 12–19.
17. Kara, L.B. and Stahovich, T.F. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers and Graphics* 29, 4 (2005), 501–517.
18. Kumar, S., Cohen, P.R., and Coulston, R. Multimodal interaction under exerted conditions in a natural field setting. In *Proceedings of the Sixth International Conference on Multimodal Interfaces* (State College, PA, Oct. 13–15). ACM Press, New York, 2004, 227–234.
19. MacEachren, A.M., Cai, G., Brewer, I., and Chen, J. Supporting map-based geo-collaboration through natural interfaces to large-screen display. *Cartographic Perspectives* 54 (Spring 2006), 16–34.
20. Moran, D.B., Cheyer, A.J., Julia, L.E., Martin, D.L., and Park, S. Multimodal user interfaces in the Open Agent Architecture. In *Proceedings of the Second International Conference on Intelligent User Interfaces* (Orlando, FL, Jan. 6–9). ACM Press, New York, 1997, 61–68.
21. Myers, K., Kolojchick, J., Angiolillo, C., Cummings, T., Garvey, T., Gervasio, M., Haines, W., Jones, C., Knittel, J., Morley, D., Ommert, W., and Potter, S. Learning by demonstration for military planning and decision making: A deployment story. In *Proceedings of the 23rd Innovative Applications of Artificial Intelligence Conference* (San Francisco, CA, Aug. 6–10). AAAI Press, Menlo Park, CA, 2011, 1597–1604.
22. O'Hara, K., Gonzalez, G., Sellen, A., Penney, G., Varnavas, A., Mentis, H., Criminisi, A., Corish, R., Rouncefield, M., Dastur, N., and Carrell, T. Touchless interaction in surgery. *Commun. ACM* 57, 1 (Jan. 2014), 70–77.
23. Oviatt, S.L. Multimodal interfaces. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, Revised Third Edition*, J. Jacko, Ed. Lawrence Erlbaum Associates, Mahwah, NJ, 2012, 405–430.
24. Oviatt, S.L. Taming recognition errors with a multimodal architecture. *Commun. ACM* 43, 9 (Sept. 2000), 45–51.
25. Oviatt, S.L. and Cohen, P.R. Perceptual user interfaces: Multimodal interfaces that process what comes naturally. *Commun. ACM* 43, 3 (Mar. 2000), 45–53.
26. Oviatt, S.L. and Cohen, P.R. *The Paradigm Shift to Multimodal in Contemporary Computer Interfaces*. Morgan & Claypool Publishers, San Francisco, CA, 2015.
27. Stilman, B., Yakhnis, V., and Umanskiy, O. Strategies in large-scale problems. In *Adversarial Reasoning: Computational Approaches to Reading the Opponent's Mind*. A. Kott and W. McEneaney, Eds. Chapman & Hall/CRC, London, U.K., 2007, 251–285.
28. U.S. Army. *U.S. Army Field Manual 101–5-1*, Chapter 5, 1997; http://armypubs.army.mil/doctrine/dr_pubs/dr_a/pdf/fm1_02c1.pdf

Philip R. Cohen (philcohen86@gmail.com) is a co-founder of Adapx, a fellow of the Association for the Advancement of Artificial Intelligence, and a past-president of the Association for Computational Linguistics.

Edward C. Kaiser (ekaiser@sensoryinc.com) is a senior application engineer at Sensory Inc., Portland, OR, and was co-PI/PT (2008–2009/2010) for STP at Adapx Inc., Seattle, WA.

M. Cecelia Buchanan (mcbuchanan@gmail.com) is a consultant at Tuatara Consulting, Seattle, WA, and was a research scientist at Adapx, Seattle, WA, when this article was written.

Scott Lind (scott.lind@adapx.com) is vice president for Department of Defense and federal solutions at Adapx, Seattle, WA.

Michael J. Corrigan (michael.corrigan@adapx.com) is a research software engineer at Adapx, Seattle, WA.

R. Matthews Wesson (matt.wesson@adapx.com) is a senior research programmer at Adapx, Seattle, WA.



Watch the authors discuss this work in this exclusive Communications video.

© 2015 ACM 0001-0782/15/04 \$15.00

DOI:10.1145/2699417

Engineers use TLA+ to prevent serious but subtle bugs from reaching production.

BY CHRIS NEWCOMBE, TIM RATH, FAN ZHANG, BOGDAN MUNTEANU, MARC BROOKER, AND MICHAEL DEARDEUFF

How Amazon Web Services Uses Formal Methods

SINCE 2011, ENGINEERS at Amazon Web Services (AWS) have used formal specification and model checking to help solve difficult design problems in critical systems. Here, we describe our motivation and experience, what has worked well in our problem domain, and what has not. When discussing personal experience we refer to the authors by their initials.

At AWS we strive to build services that are simple for customers to use. External simplicity is built on a hidden substrate of complex distributed systems. Such complex internals are required to achieve high availability while running on cost-efficient infrastructure and cope with relentless business growth. As an example of this growth, in 2006, AWS launched S3, its Simple Storage Service. In the following six years, S3 grew to store one trillion objects.³ Less than a year later it had grown to two trillion objects and was regularly handling 1.1 million requests per second.⁴

S3 is just one of many AWS services that store and process data our customers have entrusted to us. To safeguard that data, the core of each service relies on fault-tolerant distributed algorithms for replication, consistency, concurrency control, auto-scaling, load balancing, and other coordination tasks. There are many such algorithms in the literature, but combining them into a cohesive system is a challenge, as the algorithms must usually be modified to interact properly in a real-world system. In addition, we have found it necessary to invent algorithms of our own. We work hard to avoid unnecessary complexity, but the essential complexity of the task remains high.

Complexity increases the probability of human error in design, code, and operations. Errors in the core of the system could cause loss or corruption of data, or violate other interface contracts on which our customers depend. So, before launching a service, we need to reach extremely high confidence that the core of the system is correct. We have found the standard verification techniques in industry are necessary but not sufficient. We routinely use deep design reviews, code reviews, static code analysis, stress testing, and fault-injection testing but still find that subtle bugs can hide in complex concurrent fault-tolerant systems. One reason they do is that human intuition is poor at estimating the true probability of supposedly “extremely rare” combinations of events in systems operating at a scale of millions of requests per second.

» key insights

- Formal methods find bugs in system designs that cannot be found through any other technique we know of.
- Formal methods are surprisingly feasible for mainstream software development and give good return on investment.
- At Amazon, formal methods are routinely applied to the design of complex real-world software, including public cloud services.



NASA's C. Michael Holloway says, "To a first approximation, we can say that accidents are almost always the result of incorrect estimates of the likelihood of one or more things."⁸ Human fallibility means some of the more subtle, dangerous bugs turn out to be errors in design; the code faithfully implements the intended design, but the design fails to correctly handle a particular "rare" scenario. We have found that testing the code is inadequate as a method for finding subtle errors in design, as the number of reachable states of the code is astronomical. So we look for a better approach.

Precise Designs

In order to find subtle bugs in a system design, it is necessary to have a precise description of that design. There are at least two major benefits to writing a precise design: the author is forced to think more clearly, helping eliminate "plausible hand waving," and tools can be applied to check for errors in the design, even while it is being written. In contrast, conventional design documents consist of prose, static diagrams, and perhaps pseudo-code in

an ad hoc untestable language. Such descriptions are far from precise; they are often ambiguous or missing critical aspects (such as partial failure or the granularity of concurrency). At the other end of the spectrum, the final executable code is unambiguous but contains an overwhelming amount of detail. We had to be able to capture the essence of a design in a few hundred lines of precise description. As our designs are unavoidably complex, we needed a highly expressive language, far above the level of code, but with precise semantics. That expressivity must cover real-world concurrency and fault tolerance. And, as we wish to build services quickly, we wanted a language that is simple to learn and apply, avoiding esoteric concepts. We also very much wanted an existing ecosystem of tools. We were thus looking for an off-the-shelf method with high return on investment.

We found what we were looking for in TLA+,¹¹ a formal specification language based on simple discrete math, or basic set theory and predicates, with which all engineers are familiar. A TLA+ specification describes the set

of all possible legal behaviors, or execution traces, of a system. We found it helpful that the same language is used to describe both the desired correctness properties of the system (the "what") and the design of the system (the "how"). In TLA+, correctness properties and system designs are just steps on a ladder of abstraction, with correctness properties occupying higher levels, systems designs and algorithms in the middle, and executable code and hardware at the lower levels. TLA+ is intended to make it as easy as possible to show a system design correctly implements the desired correctness properties, through either conventional mathematical reasoning or tools like the TLC model checker⁹ that take a TLA+ specification and exhaustively checks the desired correctness properties across all possible execution traces. The ladder of abstraction also helps designers manage the complexity of real-world systems; designers may choose to describe the system at several "middle" levels of abstraction, with each lower level serving a different purpose (such as to understand the consequences of fin-

er-grain concurrency or more detailed behavior of a communication medium). The designer can then verify that each level is correct with respect to a higher level. The freedom to choose and adjust levels of abstraction makes TLA+ extremely flexible.

At first, the syntax and idioms of TLA+ are somewhat unfamiliar to programmers. Fortunately, TLA+ is accompanied by a second language called PlusCal that is closer to a C-style programming language but much more expressive, as it uses TLA+ for expressions and values. PlusCal is intended to be a direct replacement for pseudo-code. Several engineers at Amazon have found they are more productive using PlusCal than they are using TLA+. However, in other cases, the additional flexibility of plain TLA+ has been very useful. For many designs the choice is a matter of taste, as PlusCal is automatically translated to TLA+ with a single key press. PlusCal users do have to be familiar with TLA+ in order to write rich expressions and because it is often helpful to read the TLA+ translation to understand the precise semantics of a piece of code. Moreover, tools (such as the TLC model checker) work at the TLA+ level.

Formal Methods for Real-World Systems

In industry, formal methods have a reputation for requiring a huge amount of training and effort to verify a tiny piece of relatively straightforward code, so the return on investment is justified only in safety-critical domains (such as medical systems and avionics). Our experience with TLA+ shows this perception to be wrong. At the time of this writing, Amazon engineers have used TLA+ on 10 large complex real-world systems. In each, TLA+ has added significant value, either finding subtle bugs we are sure we would not have found by other means, or giving us enough understanding and confidence to make aggressive performance optimizations without sacrificing correctness. Amazon now has seven teams using TLA+, with encouragement from senior management and technical leadership. Engineers from entry level to principal have been able to learn TLA+ from scratch and get useful results in two to three weeks, in some



A precise, testable description of a system becomes a what-if tool for designs, analogous to how spreadsheets are a what-if tool for financial models.



cases in their personal time on weekends and evenings, without further help or training.

In this article, we have not included snippets of specifications because their unfamiliar syntax can be off-putting to potential new users. We find that potential new users benefit from hearing about the value of formal methods in industry before tackling tutorials and examples. We refer readers to Lamport et al.¹¹ for tutorials, Lamport's Viewpoint on page 38 in this issue, and Lamport¹³ for an example of a TLA+ specification from industry similar in size and complexity to some of the larger specifications at Amazon (see the table here). We find TLA+ to be effective in our problem domain, but there are many other formal specification languages and tools, some of which we describe later.

Side Benefit

TLA+ has been helping us shift to a better way of designing systems. Engineers naturally focus on designing the “happy case” for a system, or the processing path in which no errors occur. This is understandable, as the happy case is by far the most common case. That code path must solve the customer's problem, perform well, make efficient use of resources, and scale with the business—all significant challenges in their own right. When the design for the happy case is done, the engineer then tries to think of “what could go wrong” based on personal experience and that of colleagues and reviewers. The engineer then adds mitigations for these scenarios, prioritized by intuition and perhaps statistics on the probability of occurrence. Almost always, the engineer stops well short of handling “extremely rare” combinations of events, as there are too many such scenarios to imagine.

In contrast, when using formal specification we begin by stating precisely “what needs to go right.” We first specify what the system should do by defining correctness properties, which come in two varieties:

Safety. What the system is *allowed* to do. For example, at all times, all committed data is present and correct, or equivalently; at no time can the system have lost or corrupted any committed data; and

Liveness. What the system *must eventually* do. For example, whenever the

system receives a request, it must eventually respond to that request.

After defining correctness properties, we then precisely describe an abstract version of the design, along with an abstract version of its operating environment. We express “what must go right” by explicitly specifying all properties of the environment on which the system relies. Examples of such properties might be “If a communication channel has not failed, then messages will be propagated along it,” and “If a process has not restarted, then it retains its local state, modulo any intentional modifications.” Next, with the goal of confirming our design correctly handles all dynamic events in the environment, we specify the effects of each of those possible events—network errors and repairs, disk errors, process crashes and restarts, data-center failures and repairs, and actions by human operators. We then use the model checker to verify that the specification of the system in its environment implements the chosen correctness properties, despite any combination or interleaving of events in the operating environment. We find this rigorous “what needs to go right” approach to be significantly less error prone than the ad hoc “what might go wrong” approach.

More Side Benefits

We also find that writing a formal specification pays dividends over the lifetime of the system. All production services at Amazon are under constant development, even those released years ago; we add new features customers have requested, we redesign components to handle massive increases in scale, and we improve performance by removing bottlenecks. Many of these changes are complex and must be made to the running system with no downtime. Our first priority is always to avoid causing bugs in a production system, so we often have to answer “Is this change safe?” We find a major benefit of having a precise, testable model of the core system is that we can quickly verify that even deep changes are safe or learn they are unsafe without doing harm. In several cases, we have prevented subtle but serious bugs from reaching production. In other cases we have been able to

make innovative performance optimizations (such as removing or narrowing locks or weakening constraints on message ordering) we would not have dared to do without having model-checked those changes. A precise, testable description of a system becomes a what-if tool for designs, analogous to how spreadsheets are a what-if tool for financial models. We find that using such a tool to explore the behavior of the system can improve the designer’s understanding of the system.

In addition, a precise, testable, well-commented description of a design is an excellent form of documentation, which is important, as AWS systems have unbounded lifetimes. Over time, teams grow as the business grows, so we regularly have to bring new people up to speed on systems. This education must be effective. To avoid creating subtle bugs, we need all engineers to have the same mental model of the system and for that shared model to be accurate, precise, and complete. Engineers form mental models in various ways—talking to each other, reading design documents, reading code, and implementing bug fixes or small features. But talk and design documents can be ambiguous or incomplete, and the executable code is much too large to absorb quickly and might not precisely reflect the intended design. In contrast, a formal specification is precise, short, and can be explored and experimented on with tools.

What Formal Specification Is Not Good For

We are concerned with two major classes of problems with large distributed systems: bugs and operator errors that cause a departure from the system’s logical intent; and surprising “sustained emergent performance degradation” of complex systems that inevitably contain feedback loops. We know how to use formal specification to find problems in the first class. However, problems in the second class can cripple a system even though no logic bug is involved. A common example is when a momentary slowdown in a server (due, perhaps, to Java garbage collection) causes timeouts to be breached on clients, causing the clients to retry requests, thus adding load to the server, and further slowdown. In such scenarios the system eventually makes progress; it is not stuck in a logical deadlock, livelock, or other cycle. But from the customer’s perspective it is effectively unavailable due to sustained unacceptable response times. TLA+ can be used to specify an upper bound on response time, as a real-time safety property. However, AWS systems are built on infrastructure—disks, operating systems, network—that does not support hard real-time scheduling or guarantees, so real-time safety properties would not be realistic. We build soft real-time systems in which very short periods of slow responses are not considered errors. However, prolonged

Applying TLA+ to some of Amazon's more complex systems.			
System	Components	Line Count (Excluding Comments)	Benefit
S3	Fault-tolerant, low-level network algorithm	804 PlusCal	Found two bugs, then others in proposed optimizations
	Background redistribution of data	645 PlusCal	Found one bug, then another in the first proposed fix
DynamoDB	Replication and group-membership system	939 TLA+	Found three bugs requiring traces of up to 35 steps
Internal distributed lock manager	EBS Volume management	102 PlusCal	Found three bugs
	Lock-free data structure	223 PlusCal	Improved confidence though failed to find a liveness bug, as liveness not checked
	Fault-tolerant replication-and-reconfiguration algorithm	318 TLA+	Found one bug and verified an aggressive optimization

severe slowdowns *are* considered errors. We do not yet know of a feasible way to model a real system that would enable tools to predict such emergent behavior. We use other techniques to mitigate these risks.

First Steps to Formal Methods

With hindsight, Amazon's path to formal methods seems straightforward; we had an engineering problem and found a solution. Reality was somewhat different. The effort began with author C.N.'s dissatisfaction with the quality of several distributed systems he had designed and reviewed, and with the development process and tools that had been used to construct those systems. The systems were considered successful, yet bugs and operational problems persisted. To mitigate the problems, the systems used well-proven methods—pervasive contract assertions enabled in production—to detect symptoms of bugs, and mechanisms (such as “recovery-oriented computing”²⁰) to attempt to minimize the impact when bugs are triggered. However, reactive mechanisms cannot recover from the class of bugs that cause permanent damage to customer data; we must instead prevent such bugs from being created.

When looking for techniques to prevent bugs, C.N. did not initially consider formal methods, due to the pervasive view that they are suitable for only tiny problems and give very low return on investment. Overcoming the bias against formal methods required evidence they work on real-world systems. This evidence was provided by Zave,²² who used a language called Alloy to find serious bugs in the membership protocol of a distributed system called Chord. Chord was designed by an expert group at MIT and is successful, having won a “10-year test of time” award at the SIGCOMM 2011 conference and influenced several systems in industry. Zave's success motivated C.N. to perform an evaluation of Alloy by writing and model checking a moderately large Alloy specification of a non-trivial concurrent algorithm.¹⁸ We liked many characteristics of the Alloy language, including its emphasis on “execution traces” of abstract system states composed of sets and relations. However, we also found that Alloy is not expressive enough for many use cases

at AWS; for instance, we could not find a practical way in Alloy to represent rich data structures (such as dynamic sequences containing nested records with multiple fields).

Alloy's limited expressivity appears to be a consequence of the particular approach to analysis taken by the Alloy Analyzer tool. The limitations do not seem to be caused by Alloy's conceptual model (“execution traces” over system states). This hypothesis motivated C.N. to look for a language with a similar conceptual model but with richer constructs for describing system states. C.N. eventually stumbled on a language with those properties when he found a TLA+ specification in the appendix of a paper on a canonical algorithm in our problem domain—the Paxos consensus algorithm.¹²

The fact that TLA+ was created by the designer of such a widely used algorithm gave us some confidence that TLA+ would work for real-world systems. We became more confident when we learned a team of engineers at DEC/Compaq had used TLA+ to specify and verify some intricate cache-coherency protocols for the Alpha series of multicore CPUs.^{5,16} We read one of the specifications¹³ and found they were sophisticated distributed algorithms involving rich message passing, fine-grain concurrency, and complex correctness properties. That left only the question of whether TLA+ could handle real-world failure modes. (The Alpha cache-coherency algorithm does not consider failure.) We knew from Lamport's Fast Paxos paper¹² that TLA+ could model fault tolerance at a high level of abstraction and were further convinced when we found other papers showing TLA+ could model lower-level failures.¹⁵

C.N. evaluated TLA+ by writing a specification of the same non-trivial concurrent algorithm he had written in Alloy.¹⁸ Both Alloy and TLA+ were able to handle the problem, but the comparison revealed that TLA+ is much more expressive than Alloy. This difference is important in practice; several of the real-world specifications we have written in TLA+ would have been infeasible in Alloy. We initially had the opposite concern about TLA+; it is so expressive that no model checker can hope to evaluate everything that can be ex-

pressed in the language. But so far we have always been able to find a way to express our intent in a way that is clear, direct, and can be model checked.

After evaluating Alloy and TLA+, C.N. tried to persuade colleagues at Amazon to adopt TLA+. However, engineers have almost no spare time for such things, unless compelled by need. Fortunately, a need was about to arise.

First Big Success at Amazon

In January 2012, Amazon launched DynamoDB, a scalable high-performance “no SQL” data store that replicates customer data across multiple data centers while promising strong consistency.² This combination of requirements leads to a large, complex system.

The replication and fault-tolerance mechanisms in DynamoDB were created by author T.R. To verify correctness of the production code, T.R. performed extensive fault-injection testing using a simulated network layer to control message loss, duplication, and reordering. The system was also stress tested for long periods on real hardware under many different workloads. We know such testing is absolutely necessary but can still fail to uncover subtle flaws in design. To verify the design of DynamoDB, T.R. wrote detailed informal proofs of correctness that did indeed find several bugs in early versions of the design. However, we have also learned that conventional informal proofs can miss very subtle problems.¹⁴ To achieve the highest level of confidence in the design, T.R. chose TLA+.

T.R. learned TLA+ and wrote a detailed specification of these components in a couple of weeks. To model-check the specification, we used the distributed version of the TLC model checker running on a cluster of 10 cc1.4xlarge EC2 instances, each with eight cores plus hyperthreads and 23GB of RAM. The model checker verified that a small, complicated part of the algorithm worked as expected for a sufficiently large instance of the system to give high confidence it is correct. T.R. then checked the broader fault-tolerant algorithm. This time the model checker found a bug that could lead to losing data if a particular sequence of failures and recovery steps would be interleaved with other processing. This was a very subtle bug; the

shortest error trace exhibiting the bug included 35 high-level steps. The improbability of such compound events is not a defense against such bugs; historically, AWS engineers have observed many combinations of events at least as complicated as those that could trigger this bug. The bug had passed unnoticed through extensive design reviews, code reviews, and testing, and T.R. is convinced we would not have found it by doing more work in those conventional areas. The model checker later found two bugs in other algorithms, both serious and subtle. T.R. fixed all these bugs, and the model checker verified the resulting algorithms to a very high degree of confidence.

T.R. says that, had he known about TLA+ before starting work on DynamoDB he would have used it from the start. He believes the investment he made in writing and checking the formal TLA+ specifications was more reliable and less time consuming than the work he put into writing and checking his informal proofs. Using TLA+ in place of traditional proof writing would thus likely have improved time to market, in addition to achieving greater confidence in the system's correctness.

After DynamoDB was launched, T.R. worked on a new feature to allow data to be migrated between data centers. As he already had the specification for the existing replication algorithm, T.R. was able to quickly incorporate this new feature into the specification. The model checker found the initial design would have introduced a subtle bug, but it was easy to fix, and the model checker verified the resulting algorithm to the necessary level of confidence. T.R. continues to use TLA+ and model checking to verify changes to the design for both optimizations and new features.

Persuading More Engineers

Success with DynamoDB gave us enough evidence to present TLA+ to the broader engineering community at Amazon. This raised a challenge—how to convey the purpose and benefits of formal methods to an audience of software engineers. Engineers think in terms of debugging rather than “verification,” so we called the presentation “Debugging Designs.”¹⁸ Continuing the metaphor, we have found that soft-



Formal methods have helped us devise aggressive optimizations to complex algorithms without sacrificing quality.



ware engineers more readily grasp the concept and practical value of TLA+ if we dub it “exhaustively testable pseudo-code.” We initially avoid the words “formal,” “verification,” and “proof” due to the widespread view that formal methods are impractical. We also initially avoid mentioning what TLA stands for, as doing so would give an incorrect impression of complexity.

Immediately after seeing the presentation, a team working on S3 asked for help using TLA+ to verify a new fault-tolerant network algorithm. The documentation for the algorithm consisted of many large, complicated state-machine diagrams. To check the state machine, the team had been considering writing a Java program to brute-force explore possible executions: essentially a hard-wired form of model checking. They were able to avoid the effort by using TLA+ instead. Author F.Z. wrote two versions of the spec over a couple of weeks. For this particular problem, F.Z. found that she was more productive in PlusCal than TLA+, and we have observed that engineers often find it easier to begin with PlusCal.

Model checking revealed two subtle bugs in the algorithm and allowed F.Z. to verify fixes for both. F.Z. then used the spec to experiment with the design, adding new features and optimizations. The model checker quickly revealed that some of these changes would have introduced bugs.

This success led AWS management to advocate TLA+ to other teams working on S3. Engineers from those teams wrote specs for two additional critical algorithms and for one new feature. F.Z. helped teach them how to write their first specs. We find it encouraging that TLA+ can be taught by engineers who are still new to it themselves; this is important for quickly scaling adoption in an organization as large as Amazon.

Author B.M. was one such engineer. His first spec was for an algorithm known to contain a subtle bug. The bug had passed unnoticed through multiple design reviews and code reviews and had surfaced only after months of testing. B.M. spent two weeks learning TLA+ and writing the spec. Using it, the TLC model checker found the bug in seconds. The team had already designed and reviewed a fix for the bug,

so B.M. changed the spec to include the proposed fix. The model checker found the problem still occurred in a different execution trace. A stronger fix was proposed, and the model checker verified the second fix. B.M. later wrote another spec for a different algorithm. That spec did not uncover any bugs but did uncover several important ambiguities in the documentation for the algorithm the spec helped resolve.

Somewhat independently, after seeing internal presentations about TLA+, authors M.B. and M.D. taught themselves PlusCal and TLA+ and started using them on their respective projects without further persuasion or assistance. M.B. used PlusCal to find three bugs and wrote a public blog about his personal experiments with TLA+ outside of Amazon.⁷ M.D. used PlusCal to check a lock-free concurrent algorithm and then used TLA+ to find a critical bug in one of AWS's most important new distributed algorithms. M.D. also developed a fix for the bug and verified the fix. Independently, C.N. wrote a spec for the same algorithm that was quite different in style from the spec written by M.D., but both found the same bug in the algorithm. This suggests the benefits of using TLA+ are quite robust to variations among engineers. Both specs were later used to verify that a crucial optimization to the algorithm did not introduce any bugs.

Engineers at Amazon continue to use TLA+, adopting the practice of first writing a conventional prose-design document, then incrementally refining parts of it into PlusCal or TLA+. This method often yields important insight about the design, even without going as far as full specification or model checking. In one case, C.N. refined a prose design of a fault-tolerant replication system that had been designed by another Amazon engineer. C.N. wrote and model checked specifications at two levels of concurrency; these specifications helped him understand the design well enough to propose a major protocol optimization that radically reduced write-latency in the system. We have also discovered that TLA+ is an excellent tool for data modeling, as when designing the schema for a relational or “no SQL” database. We used TLA+ to design a non-trivial schema with semantic invariants over



Executive management actively encourages teams to write TLA+ specs for new features and other significant design changes.



the data that were much richer than standard multiplicity constraints and foreign key constraints. We then added high-level specifications of some of the main operations on the data that helped us correct and refine the schema. This result suggests a data model can be viewed as just another level of abstraction of the entire system. It also suggests TLA+ may help designers improve a system's scalability. In order to remove scalability bottlenecks, designers often break atomic transactions into finer-grain operations chained together through asynchronous workflows; TLA+ can help explore the consequences of such changes with respect to isolation and consistency.

Most Frequently Asked Question

On learning about TLA+, engineers usually ask, “How do we know that the executable code correctly implements the verified design?” The answer is we do not know. Despite this, formal methods still help in multiple ways:

Get design right. Formal methods help engineers get the design right, which is a necessary first step toward getting the code right. If the design is broken, then the code is almost certainly broken, as mistakes during coding are extremely unlikely to compensate for mistakes in design. Worse, engineers are likely to be deceived into believing the code is “correct” because it appears to correctly implement the (broken) design. Engineers are unlikely to realize the design is incorrect while focused on coding;

Gain better understanding. Formal methods help engineers gain a better understanding of the design. Improved understanding can only increase the chances they will get the code right; and

Write better code. Formal methods can help engineers write better “self-diagnosing code” in the form of assertions. Independent evidence¹⁰ and our own experience suggest pervasive use of assertions is a good way to reduce errors in code. An assertion checks a small, local part of an overall system invariant. A good system invariant captures the fundamental reason the system works; the system will not do anything wrong that could violate a safety property as long as it continuously maintains the system invariant.

The challenge is to find a good system invariant, one strong enough to ensure no safety properties are violated. Formal methods help engineers find strong invariants, so formal methods can help improve assertions that help improve the quality of code.

While we would like to verify that executable code correctly implements the high-level specification or even generate the code from the specification, we are not aware of any such tools that can handle distributed systems as large and complex as those being built at Amazon. We do routinely use conventional static analysis tools, but they are largely limited to finding “local” issues in the code, and are unable to verify compliance with a high-level specification.

We have seen research on using the TLC model checker to find “edge cases” in the design on which to test the code,²¹ an approach that seems promising. However, Tasiran et al.²¹ covered hardware design, and we have not yet tried to apply the method to software.

Alternatives to TLA+

There are many formal specification methods. We evaluated several and published our findings in Newcombe,¹⁹ listing the requirements we think are important for a formal method to be successful in our industry segment. When we found TLA+ met those requirements, we stopped evaluating methods, as our goal was always practical engineering rather than an exhaustive survey.

Related Work

We find relatively little published literature on using high-level formal specification for verifying the design of complex distributed systems in industry. The Farsite project⁶ is complex but somewhat different from the types of systems we describe here and apparently never launched commercially. Abrial¹ cited applications in commercial safety-critical control systems, but they seem less complex than our problem domain. Lu et al.¹⁷ described post-facto verification of a well-known algorithm for a fault-tolerant distributed hash table, and Zave²² described another such algorithm, but we do not know if these algorithms have been used in commercial products.

Conclusion

Formal methods are a big success at AWS, helping us prevent subtle but serious bugs from reaching production, bugs we would not have found through any other technique. They have helped us devise aggressive optimizations to complex algorithms without sacrificing quality. At the time of this writing, seven Amazon teams have used TLA+, all finding value in doing so, and more Amazon teams are starting to use it. Using TLA+ will improve both time-to-market and quality of our systems. Executive management actively encourages teams to write TLA+ specs for new features and other significant design changes. In annual planning, managers now allocate engineering time to TLA+.

While our results are encouraging, some important caveats remain. Formal methods deal with models of systems, not the systems themselves, so the adage “All models are wrong, some are useful” applies. The designer must ensure the model captures the significant aspects of the real system. Achieving it is a special skill, the acquisition of which requires thoughtful practice. Also, we were solely concerned with obtaining practical benefits in our particular problem domain and have not attempted a comprehensive survey. Therefore, mileage may vary with other tools or in other problem domains. ■

References

1. Abrial, J. Formal methods in industry: Achievements, problems, future. In *Proceedings of the 28th International Conference on Software Engineering* (Shanghai, China, 2006), 761–768.
2. Amazon.com. *Supported Operations in DynamoDB: Strongly Consistent Reads*. System documentation; <http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/APISummary.html>
3. Barr, J. Amazon S3: The first trillion objects. Amazon Web Services Blog, June 2012; <http://aws.typepad.com/aws/2012/06/amazon-s3-the-first-trillion-objects.html>
4. Barr, J. Amazon S3: Two trillion objects, 1.1 million requests per second. Amazon Web Services Blog, Mar. 2013; <http://aws.typepad.com/aws/2013/04/amazon-s3-two-trillion-objects-11-million-requests-second.html>
5. Batson, B. and Lamport, L. High-level specifications: Lessons from industry. In *Formal Methods for Components and Objects, Lecture Notes in Computer Science Number 2852*, F.S. de Boer, M. Bonsangue, S. Graf, and W.-P. de Rover, Eds. Springer, 2003, 242–262.
6. Bolosky, W., Douceur, J., and Howell, J. The Farsite Project: A retrospective. *ACM SIGOPS Operating Systems Review: Systems Work at Microsoft Research* 41, 2 (Apr. 2007), 17–26.
7. Brooker, M. Exploring TLA+ with two-phase commit. Personal blog, Jan. 2013; <http://brooker.co.za/blog/2013/01/20/two-phase.html>
8. Holloway, C. Michael Why you should read accident reports. Presented at the *Software and Complex Electronic Hardware Standardization Conference*

- (Norfolk, VA, July 2005); http://klabs.org/richcontent/conferences/aa_nasa_2005/presentations/cmh-why-read-accident-reports.pdf
9. Joshi, R., Lamport, L. et al. Checking cache-coherence protocols with TLA+. *Formal Methods in System Design* 22, 2 (Mar. 2003) 125–131.
 10. Kudrjavec, G., Nagappan, N., and Ball, T. Assessing the relationship between software assertions and code quality: An empirical investigation. In *Proceedings of the 17th International Symposium on Software Reliability Engineering* (Raleigh, NC, Nov. 2006), 204–212.
 11. Lamport, L. The TLA Home Page; <http://research.microsoft.com/en-us/um/people/lamport/tla/tla.html>
 12. Lamport, L. Fast Paxos. *Distributed Computing* 19, 2 (Oct. 2006), 79–103.
 13. Lamport, L. The Wildfire Challenge Problem; <http://research.microsoft.com/en-us/um/people/lamport/tla/wildfire-challenge.html>
 14. Lamport, L. Checking a multithreaded algorithm with +CAL. In *Distributed Computing: 20th International Conference*, S. Dolev, Ed. Springer-Verlag, 2006, 11–163.
 15. Lamport, L. and Merz, S. Specifying and verifying fault-tolerant systems. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, Lecture Notes in Computer Science, Number 863*, H. Langmaack, W.-P. de Rover, and J. Vytotil, Eds. Springer-Verlag, Sept. 1994, 41–76.
 16. Lamport, L., Sharma, M., Tuttle, M., and Yu, Y. *The Wildfire Challenge Problem*. Jan. 2001; <http://research.microsoft.com/en-us/um/people/lamport/pubs/wildfire-challenge.pdf>
 17. Lu, T., Merz, S., and Weidenbach, C. Towards verification of the Pastry Protocol using TLA+. In *Proceedings of Joint 13th IFIP WG 6.1 International Conference and 30th IFIP WG 6.1 International Conference Lecture Notes in Computer Science Volume 6722* (Reykjavik, Iceland, June 6–9), Springer-Verlag, 2011, 244–258.
 18. Newcombe, C. Debugging Designs. Presented at the *14th International Workshop on High-Performance Transaction Systems* (Monterey, CA, Oct. 2011); http://hpts.ws/papers/2011/sessions_2011/Debugging.pdf and associated specifications http://hpts.ws/papers/2011/sessions_2011/amazonbundle.tar.gz
 19. Newcombe, C. Why Amazon chose TLA+. In *Proceedings of the Fourth International Conference Lecture Notes in Computer Science Volume 8477*, Y.A. Ameur and K.-D. Schewe, Eds. (Toulouse, France, June 2–6), Springer, 2014, 25–39.
 20. Patterson, D., Fox, A. et al. The Berkeley/Stanford Recovery-Oriented Computing Project. University of California, Berkeley; <http://roc.cs.berkeley.edu/>
 21. Tasiran, S., Yu, Y., Batson, B., and Kreider, S. Using formal specifications to monitor and guide simulation: Verifying the cache coherence engine of the Alpha 21364 microprocessor. In *Proceedings of the Third IEEE International Workshop on Microprocessor Test and Verification* (Austin, TX, June). IEEE Computer Society, 2002.
 22. Zave, P. Using lightweight modeling to understand Chord. *ACM SIGCOMM Computer Communication Review* 42, 2 (Apr. 2012), 49–57.

Chris Newcombe (chris.newcombe@gmail.com) is an architect at Oracle, Seattle, WA, and was a principal engineer in the AWS database services group at Amazon.com, Seattle, WA, when this article was written.

Tim Rath (rath@amazon.com) is a principal engineer in the AWS database services group at Amazon.com, Seattle, WA.

Fan Zhang (fanxhang58@gmail.com) is a software engineer and technical product and program manager at Cyanogen, Seattle, WA, and was a software engineer for AWS S3 at Amazon.com, Seattle, WA, when this article was written.

Bogdan Munteanu (bogdanmunte@gmail.com) is a software engineer at Dropbox, and was a software engineer in the AWS S3 Engines group at Amazon.com, Seattle, WA, when this article was written.

Marc Brooker (mbrooker@amazon.com) is a principal engineer for AWS EC2 at Amazon.com, Seattle, WA.

Michael Deardeuff (mdearde@amazon.com) is a software engineer in the AWS database services group at Amazon.com, Seattle, WA.

Copyright held by Owners/Authors.
Publication rights licensed to ACM. \$15.00

Implantable devices, often dependent on software, save countless lives. But how secure are they?

BY JOHANNES SAMETINGER, JERZY ROZENBLIT, ROMAN LYSECKY, AND PETER OTT

Security Challenges for Medical Devices

SECURITY AND SAFETY issues in the medical domain take many different forms. Examples range from purposely contaminated medicine to recalls of vascular stents, and health data breaches. Risks resulting from unintentional threats have long been known, for example, interference from electromagnetic energy.

Security risks resulting from intentional threats have only recently been confirmed, as medical devices increasingly use newer technologies such as wireless communication and Internet access. Intentional threats include unauthorized access of a medical device or unauthorized change of settings of such a device. A senior official in the device unit of the U.S. Food and Drug Administration (FDA) has often been cited with the following statement: “We are aware

of hundreds of medical devices that have been infected by malware.”³⁴ Even though deaths and injuries have not yet been reported from such intrusions, it is not difficult to imagine that someday they will. There is no doubt that health care will increasingly be digitized in the future. Medical devices will increasingly become smarter and more interconnected. The risk of computer viruses in hospitals and clinics is one side effect of this trend. Without suitable countermeasures, more data breaches and even malicious attacks threatening the lives of patients may result.

Security is about protecting information and information systems from unauthorized access and use. As mentioned, medical devices have more and more embedded software with communication mechanisms that now qualify them as information systems. Confidentiality, integrity, and availability of information are core design and operational goals. Secure software is supposed to continue to function correctly under a malicious attack.²⁵ In this sense, medical device security is the idea of engineering these devices so they continue to function correctly even if under a malicious attack. This includes internal hardware and software aspects as well as intentional and unintentional external threats.

Medical devices comprise a broad range of instruments and implements.

» key insights

- Healthcare poses security challenges due to the sensitivity of health records, the increasing interoperability of medical devices, and simply the fact that human well-being and life are at stake.
- Implantable devices are especially critical, as they may potentially put patients in life-threatening situations when not properly secured.
- Medical devices are becoming noticeably important for millions of patients worldwide. Their increasing dependence on software and interoperability with other devices via wireless communication and the Internet has put security at the forefront.



IMAGE BY MOPIC

For our considerations, only devices with hardware, software, and some form of interoperability are of interest. Artificial joints, for example, do not do any processing, that is, there is no software involved. Thus, we can ignore them from a security perspective. However, they may indeed be critical from a safety point of view.

At this point, we emphasize the importance of secure medical devices. It is not really about preventing someone from killing someone else by means of a medical device. However remote and unlikely this scenario might sound, it is not completely implausible. Securing medical devices

is securing a critical infrastructure. It is about preventing malicious people from taking control of this infrastructure, about preventing a potential blackmail of device manufacturers or health institutions, and about the sense of well-being of any person who needs to use any such device.

Motivation

Major IT security incidents that affect the general public are almost regularly reported in the media. Examples include stolen passwords, stolen credit card information, or website availability problems. The loss, theft, or exposure of personally identifiable

information is one major problem that is also widespread in the health care sector, which accounts for one-fifth of all these reported issues.³³ The FDA collects information regarding reportable issues with medical devices to capture and identify adverse and unexpected events for a particular device or device type. Each year, several hundred thousand medical device reports are received about suspected device-associated deaths, serious injuries, and malfunctions.⁶ An analysis of these recalls and events has shown that both the number of recalls and adverse events have increased over the years.

The major reason for device recalls involves malfunctions. Computer-related recalls account for about 20% to 25%, and counting. The numbers show that computer-related recalls are caused mainly by software.¹ More than 90% of device recalls mentioned the word ‘software’ as the reason for the corrective action. Less than 3% mentioned an upgrade would be available online.²³ Kramer et al. also tested the FDA’s adverse event reporting by notifying a device’s vulnerability, only to find out that it took several months before the event showed up in the corresponding database. This time span is definitely much too long to respond to software-related malfunctions.

Successful hacking of medical devices has been demonstrated on several occasions. For example, commands have been sent wirelessly to an insulin pump (raise or lower the levels of insulin, disable it). This could be done within a distance of up to 150 feet.²⁰ The FDA’s safety communication has issued a warning to device makers and health care providers to put safeguards in place to prevent cyber-attacks.⁹ Deaths or injuries are not yet known, but the hypothetical ramifications are obvious. The non-medical IT landscape can also pose a threat to medical operations. For example, when computers around the world came to a halt after an antivirus program identified a normal system file as a virus, hospitals had to postpone elective surgeries and to stop treating patients.¹¹

Medical Devices

Medical devices include everything from simple wooden tongue depressors and stethoscopes to highly sophisticated computerized medical equipment.³⁷ According to the World Health Organization (WHO), a medical device is “an instrument, apparatus, implement, machine, contrivance, implant, in vitro reagent, or other similar or related article” intended for use in the diagnosis, prevention, monitoring, and treatment of disease or other conditions.³⁷ The FDA uses a similar definition.⁷ Classes of medical devices have been defined differently in, for example, the U.S., Canada, Europe, or Australia. The FDA has established classifica-

tions for approximately 1,700 different generic types of devices. These devices are grouped into medical specialties, called panels. Examples for the FDA’s specialty panels include cardiovascular devices, dental, orthopedic, as well as ear, nose, and throat devices. Active devices may or may not involve software, hardware, and interfaces, which are important when considering security issues. These devices can do some processing, receive inputs from outside the device (sensors), output values to the outer world (actuators), and communicate with other devices.

Device safety. Each of the FDA’s generic device types is assigned to one of three regulatory classes: I, II, and III. The classes are based on the level of control necessary to ensure the safety and effectiveness of a device; the higher the risk, the higher the class.⁸ For example, class III devices have to be approved by a premarket approval process. This class contains devices that are permanently implanted into human bodies and may be necessary to sustain life, for example, artificial hearts or an automated external defibrillator. The classification is based on the risk that a device poses to the patient or the user. Class I includes devices with the lowest risk, class III those with the greatest risk.

According to the WHO, optimum safety and performance of medical devices requires risk management with the cooperation among all involved in the device’s life span, that is, the government, the manufacturer, the importer/vendor, the user, and the public.³⁷ The international standard ISO 14971: 2007 provides a framework for medical device manufacturers including risk analysis, risk evaluation, and risk control for risk management in a device’s design, development, manufacturing, and after-sale monitoring of a device’s safety and performance.¹⁸

Device security. We consider a medical device to be security-critical if it does some form of processing and communicating, typically by running some form of software on specialized hardware, and often, employing a range of sensors.⁷ Sensing devices constitute a security threat because wrong sensor values may later induce therapeutically wrong decisions by

doctors or devices. Safety-critical information has an influence on the safety of a person or her environment. Examples include parameter settings or commands for devices such as implanted defibrillators or x-ray machines. Both malicious and unintentional modification of such information may lead to safety-critical situations. Sensitive information includes anything that is about a patient, for example, medical records as well as values from sensing devices that report information about a person’s or her device’s state, for example, glucose level, ID, or parameter settings of a pacemaker. It is interesting to note that all medical devices as defined by the WHO or by the FDA have aspects that are inherently safety related. Some have a higher risk, some a lower one (see FDA’s classes I, II, and III). However, not all of these devices are relevant from a security point of view; recall the aforementioned artificial joint. Typically, security is an issue as soon as software is involved. But there are also security-relevant devices that are not considered to be medical devices by the WHO or the FDA. Examples include smartphones that run medical apps handling sensitive information, or regular PCs in a hospital for processing medical records.

The difference between safety and security is not always obvious because security can clearly have an effect on safety. Generally speaking, safety is about the protection of a device’s environment, that is, mainly the patient, from the device itself. The manufacturer must ensure the device does not harm the patient, for example, by not using toxic substances in implants or by careful development of an insulin pump’s software. Security is about the protection of the device from its environment, that is, just the opposite of safety. As long as a device is operating in a stand-alone mode, this is not an issue. But if a device communicates with its environment or is connected to the Internet or other systems, then someone may get access to data on the device or even gain control over it. A security issue becomes a safety issue when a malicious attacker gains control of a device and harms the patient.

Non-communicating but processing devices can be critical to security

when attackers have managed to implant malicious hardware or software before the device gets installed. Examples include hardware or software Trojans that might be installed in heart pacemakers to be activated upon a specific event. Precautions must be taken at the design and development processes in order to avoid such attacks. Communicating devices, of course, provide a broader attack “surface.”

We suggest a security classification of medical devices depending on whether they process or communicate sensitive information and on whether they process or communicate safety-critical information. The accompanying table summarizes our proposed levels for devices that are security-relevant. Note this set is an initial classification. While not yet fully elaborated, it is a first step toward developing a more comprehensive taxonomy of security levels.

Health care professionals increasingly improve and facilitate patient care with mobile medical applications. An increasing number of patients manage their health and wellness with such applications. Such apps may promote healthy living and provide access to useful health information. Mobile medical apps can be used for a plethora of uses. They can extend medical devices by connecting to them for the purpose of displaying, storing, analyzing, or transmitting patient-specific data. Not every mobile medical application necessarily poses a security risk. However, as soon as it processes or transmits sensitive information or even controls the medical device, security precautions must be taken.

Pacemaker Scenario

We will illustrate security issues through an example of pacemakers, that is, medical devices that are implanted in patients to regulate the patient’s heart rate. The purpose of such a device is to maintain an adequate heart rate of a patient whose heart would not be able to do so otherwise. Pacemakers are classified as Class III, the highest safety category.

Clinical perspective. Implantable medical devices are prevalent in many medical specialties. The implantable cardiac pacemakers and defibrilla-

tors can be especially critical for the patient’s health and welfare. These devices are implanted in hundreds of thousands of patients every year; many of these patients would not be able to live without a fully functional device. Patients with these types of implantable devices are typically seen in a follow-up on a regular basis, in an outpatient clinic or hospital setting, where the device is interrogated and adjustments are made as needed. Trained staff or physicians perform these functions using a vendor-specific programming system, which communicates with the device by means of a wand or wireless technology. In addition, over the last several years essentially all device vendors have established a home-based device follow-up system. For this purpose, a data module is located at the patient’s home, typically at the bedside. Once the patient is in proximity to the data module, wireless contact is established and the data module interrogates the device. This information is sent (typically through a telephone landline) to an Internet-based repository. Authorized health care professionals can view this information.

Implantable cardiac pacemakers and defibrillators are highly reliable. Nevertheless, failure of device components has occurred and highlighted the potential medical and legal implications. These failures have largely been due to problems with manufacturing processes and/or materials and have typically been limited to certain device batches. Almost always, however, such device failures require surgical device replacement. With the increasing prevalence of Web-based wireless remote device follow-up systems, concerns about device security

have arisen. At this time these remote follow-up systems are in read-only mode. However, device programming through remote follow-up systems is being investigated. Incorrect programming either by error, technical failure, or malicious intent could have potentially life-threatening implications for the patient.

Risk assessment. In our pacemaker scenario, we distinguish different risks according to the CIA triad, confidentiality, integrity, and availability. First—confidentiality—sensitive data about the patient and her pacemaker may be disclosed. Second—integrity—data on a device may be altered, resulting in a range of slightly to highly severe impacts on the patient. Third—availability—may render a device inoperable. An architectural overview of the pacemaker environment is given in the accompanying figure on page 79. While the pacemaker itself is communicating wirelessly, other communication is done via the Internet, a phone line, and sometimes by means of a USB stick. Even if programming devices may not yet have a direct connection to the clinic, sooner or later, they will.

Information disclosure and tampering may happen on any connection between devices. On the Internet, a man-in-the-middle attack can occur, unless appropriate measures such as encryption mechanisms have been used. Wireless communication additionally allows attackers to listen to the traffic with a separate device, that is, another programming device, another home monitor, or a different device specifically for an attack. Such devices can be used not only for listening but also to pretend being an authorized communication partner. Denial-of-service attacks may occur as well. In our sce-

Security levels of medical devices.

Security level	Description	Device examples
<i>Low</i>	Neither sensitive nor safety-critical activity	PC in hospital used for administrative work Heart rate watch
<i>Medium</i>	Sensitive activity	PC processing electronic health records (EHRs) Smartphone communicating glucose levels
<i>High</i>	Safety-critical activity	Device controlling insulin pump or sending parameters to pacemaker
<i>Very High</i>	Safety-critical activity, input from elsewhere	Pacemaker receiving external parameters

nario, the biggest threat stems from the pacemaker's interoperability. The purpose of an assessment of a device's risks is a determination of risks, their degree of harm as well as the likelihood of harm occurring.²⁷ Based on this information, countermeasures must be identified and selected.

Software. Vulnerabilities in software are bugs or flaws in software that can directly be used by attackers to gain access to a system or network. Software for pacemakers is confidential and proprietary. A system specification is available for academic purposes.² It demonstrates the complexity of these seemingly simple devices. There are many programmable parameters, for example, lower and upper rate limit, as well as various delays and periods. Functionality includes device monitoring, lead support, pulse pacing, various operation modes and states as well as extensive diagnostic features. Software is not only needed on the pacemaker itself, but also on the programming device and on the home monitor. Software on the programming device is needed to non-invasively reprogram a pacemaker, for example, to modify the pacemaker rate, to monitor specific functions, and to process data obtained from the pacemaker. Such software can work with one or a few models of devices, typically from the same manufacturer. Software on the home monitor has to communicate with the pacemaker and to mainly upload important information to a specific server, where personnel from the clinic can later access it. Installing updates may be necessary on both the programming and the home monitor, but also on the pacemaker itself. A compromised pacemaker can directly do harm to its patient. A compromised programming device can do so indirectly. It may just send other parameters to the device than the ones the cardiologist has chosen. A compromised home monitor also poses a serious threat. If it uploads incorrect values to the server, then these values may lead the cardiologist to wrong conclusions and eventually to wrong device settings that may harm the patient. Last but not least, a compromised server that stores all these values poses a similar threat.

Hardware. Hidden malicious circuits provide attackers with stealthy



Medical device security is the idea of engineering these devices so they continue to function correctly even if under a malicious attack.



attack vectors.²¹ Various potential attacks like privilege escalation, login backdoor, and password stealing have been demonstrated. The hardware of pacemakers is, like its software, confidential and proprietary. A hardware reference platform is available at the University of Minnesota. It is based upon an 8-bit microcontroller.²⁶ Hardware for programming devices and home monitors is less constrained.

These devices have no space and power constraints and are comparable to regular PCs. Similarly to software, malicious hardware circuits can be placed on the medical device itself, but also on other devices it communicates with, such as the programming device and the home monitor in our pacemaker scenario. Malicious hardware on the Web server, where pacemaker data is stored, also poses a threat by either revealing sensitive medical data or by even modifying this data and, thus, misleading the treating physician.

Interoperability. Security issues of pacemakers have also been raised due to their capability of wireless communication. Concerns include unauthorized access to patient data on the device as well as unauthorized modifications of the device's parameters.

Needless to say, modified settings may harm patients' well-being, cause severe damages to their hearts, and even cause their deaths. Device integrity is at stake when its wireless communication is attacked. The crucial question is whether it is possible for unauthorized third parties to change device settings, to change or disable therapies, or even to deliver command shocks. Halperin et al. have partially reverse engineered a pacemaker's communications protocol with an oscilloscope and a software radio and have then implemented several attacks able to compromise the safety and privacy of patients.¹⁵

Even if hardware and software of all devices in our pacemaker scenario are free of malware, an attacker may still pose a threat by communicating with either one of these devices, such as the home monitor, the programming device, the service provider's Web server, or the pacemaker itself. Interoperability requires protocols that define sequences of operations between the two communicating parties. These se-

quences must ensure the protection of data. Network protocols have often suffered from vulnerabilities, thus, allowing attackers to pretend being someone else. Attackers may use a modified programming device with stronger antennae that allow them to communicate with a pacemaker from a longer distance. They may then pretend to be the authorized cardiologist and modify settings of the device. Similarly, they may act as the home monitor and read out sensitive data, or communicate with the home monitor, pretending to be the pacemaker, and relay wrong values.

Challenges

Critical assets deserving strong protection in health care include medical records, a plethora of medical sensors and devices, and last but not least, human health and life. The security of medical devices is different and more challenging vis-à-vis regular IT security for several reasons, not just because of the fact that human life is at stake. Clearly, nonmedical devices like automobiles can also endanger human life if their safety is compromised through a security breach. One can imagine a scenario where malware is implanted into a dynamic stability control system to intentionally cause an accident. But many medical devices impact the patients' physiology and, thus, pose a permanent threat. Resource constraints are present not for all, but for many, most notably implanted medical devices. Little memory, processing power, physical size limitations and battery life limit the options that are available for security countermeasures. Emergency situations provide an additional challenge that is not present in other domains. Medical devices must prevent unauthorized access, yet may need to allow for quick and simple access in emergency situations. Another problem is reproducibility. Security researchers often lack access to proprietary devices and are, thus, limited in their ability to study attacks and defenses.

Several countermeasures to vulnerabilities in medical devices have been described.^{4,14} They can be protective, corrective, or detective. Examples are auditing, notification, trusted external or internal devices, and cryptographic protections.¹⁶ Here, we enumerate vari-

ous challenges and postulate a means of tackling them.

Software security. Besides the functionality, software developers of medical devices must take measures to ensure the safety as well as the security of their code. Both secure development and secure update mechanisms are needed. Risks of medical device software have also been described in Fu and Blum.¹²

Secure development. Security is a volatile property. A system is never 100% secure. As long as vulnerabilities are unknown, this is not a problem. When attackers know a specific vulnerability, the target system is at risk. The engineering of secure medical software is not radically different from the development of other types of software. It is a common misconception that only bad programmers write insecure code. Besides the underlying complexity of writing code, it takes detailed knowledge, extra training, and additional development activities in order to write secure code.¹⁷ Thus, economic and sometimes social factors often play against security quality.

In medical device software we must ensure both safety and security have top priority and there is a defined process to report and fix vulnerabilities. The challenge for medical devices includes the fact that additional code for security must not interfere with real-time constraints and other resource constraints like limited battery power.

Update mechanisms. When manufacturers of a system know about vulnerabilities, they will address and correct the problems. A fix must then be

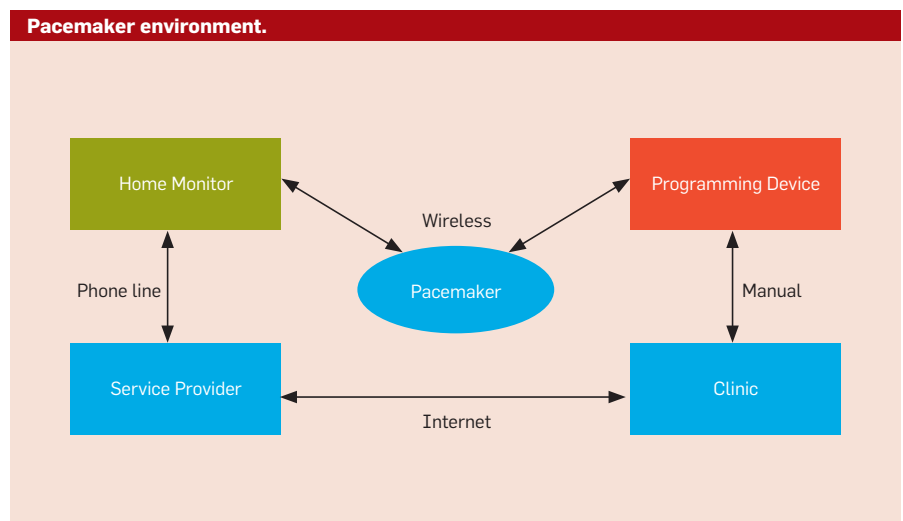
distributed to the systems with that vulnerability. The update mechanism itself may be misused for an attack. Updates and patches are (still) much less frequent for medical devices than they are for personal computers and smartphones. However, sometimes they will be necessary.

We need user-friendly update processes for medical devices and take precautions such that malware is not involved in the update process itself. In addition, the update must not break the device or halt its proper functioning.

Off-the-shelf software often “powers” medical technology. On medical devices, software patches or updates are often delayed or are even missing altogether. Missing patches may also be an organizational problem. Delays may result from the fact that device manufacturers must approve upgrades to software as well as any security installations.³⁶ The problem with old software versions is they often contain known vulnerabilities.

Old software in medical devices was not an issue as long as these devices operated stand-alone. Increasing interconnection makes these devices vulnerable even with old malware.¹² For medical devices, it is important the production life cycles of embedded software must match the devices' production life cycles. Manufacturers must ensure software is not used on medical devices after its support has expired.

Hardware security. Safety issues are more prevalent in hardware than the security concerns. An example includes the electromagnetic interference of non-medical devices with pace-



makers. Hardware Trojans on medical devices seem unrealistic today, but precautions must be taken to reduce attack vectors wherever possible. Backdoors in military chips have already been documented, where attackers could extract configuration data from the chip, reprogram crypto and access keys, modify low-level silicon features, and also permanently damage the device.³⁰ An approach for automatic embedding of customizable hardware Trojan horses into arbitrary finite state machines has been demonstrated. These Trojan horses are undetectable and improvable.³⁵ Radio pathways have been embedded into computers, where computers could be remotely controlled and provided with malware even when they were not connected to the Internet.²⁹

We must keep in mind that hardware Trojans can be an attack vector for medical devices too. It is important to ensure such malware is not installed in the manufacturing process. Given the reliance on computer-aided design tools, it is further necessary to ensure hardware Trojans are not inserted in the design by these tools. Verification methods utilized in designing hardware should ensure the resulting output designs match the inputs and do not contain additional circuitry. Outside of using trusted manufactures for each stage of design, ensuring Trojan-free hardware is not practical. Thus, detection and mitigation capabilities will still be needed. Once malicious hardware is detected and its behavior is understood, research on how to mitigate the affects of the malicious hardware to ensure safety of medical devices will be of critical importance.

Interoperability. Increasingly, medical devices rely on wireless connectivity, be it for remote monitoring, or for remote updates of settings or even for an update of the software itself. Interoperability challenges include secure protocols, authentication, authorization, encryption, and key management. Interoperability of medical devices is especially tricky due to medical emergency situations. In case of an emergency, health personnel may need to access not only medical records, but also medical devices of a person in need, perhaps in a life-threatening situation. Authentication and authorization mechanisms

must have a bypass or shortcut for such circumstances. However, these bypasses and shortcuts should not provide a means that enables attackers to gain access to the device.

Initiatives to secure the interoperability of medical devices include externally worn devices,³ for example, a trustworthy wrist-worn amulet,³¹ and software radio shields.¹³ Researchers have also created a prototype firewall to block hackers from interfering with wireless medical devices³² and to authenticate via physical contact and the comparison of ECG readings.²⁸

Organizational. Security is most effective when designed into the system from the very initial development cycle. It is important to develop and maintain threat models and to assess risks during device development. A systematic plan for the provision of software updates and patches is needed. Last but not least, a security response team has to permanently identify, monitor, and resolve security incidents and security vulnerabilities.

For that purpose, user facilities such as hospitals and clinics should be incentivized to report security occurrences. These reports can provide valuable insights into security problems of medical devices. In addition, we propose the definition of security and threat levels for medical devices with defined rules of action and an audit guideline for all involved stakeholders. The levels defined in the table here are a small first step in that direction. We imagine simple scores for medical devices that summarize their sensitivity, their impact as well as their exposure and their current threat level. Rule-based actions could then trigger needed actions to react to security-related incidents.

Regulations. It is important to know at any time the level of danger and to take appropriate countermeasures. Design and distribution of medical devices is tightly regulated. In the U.S., the FDA has the authority over medical device distribution. A device manufacturer has the responsibility for the approved configuration of the device. Device users, such as hospitals and patients, do not have access to a device's software environment and cannot install additional security measures. Any upgrade or update—either added functionality or security measures—

typically needs to be approved by the manufacturer. Thus, deployment of security-relevant upgrades typically gets delayed.³⁶ Manufacturers, importers, and device user facilities are required to report specific device-related adverse events and product problems.

Surveillance strategies must be reconsidered in order to effectively and efficiently collect data on security and privacy problems in medical devices.²³ Some regulation aspects as well as the role of standards bodies, manufacturers, and clinical facilities have been discussed in Fu and Blum.¹² We see a demand for action to adjust the increasing need for software updates for medical devices with the need to redo clinical trials after major changes.

Malware detection. Vulnerabilities are often unknown until malware exploiting those vulnerabilities is detected. We need methods to detect the presence of malware. Malware detection techniques include control-flow integrity verification, call stack monitoring, dataflow analysis, and multisource hash-based verification. Although software-based malware detection methods are suitable for traditional computing systems, the performance overhead may be prohibitive for medical devices with strict time constraints. Hardware-based detection methods can reduce or eliminate the performance overhead, but power consumption remains a challenge.

For medical devices, we need malware detection methods that are non-intrusive with very low power consumption, as power is a precious resource, especially in implantable devices. In order to provide resilience to zero-day exploits, anomaly-based malware detection methods will be needed. These methods rely on accurate models of normal system behavior, which will require both formal methods for modeling this behavior and tight integration with system design tasks. The importance of timing requirements in medical devices may provide a unique system feature that can be exploited to better detect malware.

Malware reaction. Detecting malware only addresses half of the problems. Once malware is detected, how should the medical device respond? Notification is a straightforward option, but it allows the malware to re-

main active until the device can be inspected or replaced. Automatically reinstalling the software may be feasible if halting the device temporarily is safe for the patient. We live in an interconnected world. Unplugging from the Internet may cause a bit of distress but is unlikely to harm one physically. However, life-critical medical devices present a much more complex set of challenges. Clearly, any reprogramming, resetting, or disconnecting a device such as a demand pacemaker, which paces the heart only if the rhythm is abnormal, is less disruptive than it would be in a permanent pacemaker. Trade-off decisions must be considered in such situations. Replacing the device might be an option, but what about the time until the device gets replaced? Being able to turn off any communication to the device is at least a first step, which had been taken by a former U.S. Vice President to avoid a potential terroristic attack.²² It has to be clear, though, that this step may come too late if malware had already been planted onto the device before terminating the communication capabilities. Resetting the device may be an option in this scenario.

Notifications alert patients to potentially malicious activities.¹⁵ However, notifications of security breaches would rather unnerve worried patients. We imagine different device modes that may be switched when malware is suspected or even known. One such mode, for example, could switch off any communication and use predefined, safe parameter settings. Critically, the design of alternative safe modes must ensure various software implementations are isolated, both through software safeguards and secure hardware architectures, such that malware cannot alter the operation of the safe modes. Fail-safe features must protect a device's critical functionality, even when security has been compromised.¹⁰

Formal methods. Finding vulnerabilities in software and hardware before being deployed within a medical device can significantly increase security. In practice, eliminating all security vulnerabilities is infeasible and impractical. Formal verification methods can be applied to analyze temporal behavior and to detect potential



Attackers may use a modified programming device with stronger antennae that allow them to communicate with a pacemaker from a longer distance.



vulnerabilities.²⁴ Guaranteeing timing properties is an important issue when developing safety-critical real-time systems like cardiac pacemakers. Jee et al. have presented a safety assured development approach of real-time software using a pacemaker as a case study.¹⁹ They followed model-driven development techniques and used measurement-based timing analysis to guarantee timing properties both in their implementation and in the formal model.

Formal methods play an important role in ensuring the hardware and software for medical devices operate as designed. We further believe formal methods should be utilized to verify correctness of software updates, malware reaction methods, and other runtime system reconfigurations. Formal modeling and verification are essential to ensuring changes to the system at runtime can be accomplished without impacting device behavior.

Resource constraints. Limited power/energy and limited sizes may make known security solutions impractical. For example, an implanted defibrillator may not have the resources to run commercial anti-virus software. Even if it could do so, it may drain the battery too much. In addition, such software would have to connect to the Internet to keep virus information up to date and, thus, open up yet another attack vector. Limited memory may necessitate the use of scaled back versions of operating systems. It also makes it more difficult to utilize common security software.³⁶

Recent research has shown tiny power generators that can convert the motion of a beating heart into electrical energy and implantable devices that can be wirelessly recharged. Zero-power notification and authentication with induced RF energy at no cost to the battery has also been shown, for example, to audibly alert patients of security-sensitive events.¹⁵ But limited resources will still confine security measures in many medical devices.

Non-technical aspects. In addition to technical security aspects of medical devices, we have to consider non-technical issues as well. Security awareness is one major aspect. Technical security measures are useless, when people, for example, provide login credentials to

unauthorized people. Technically viable systems may nonetheless be undesirable to patients.

The general population is increasingly concerned about the misuse of the Internet in many aspects of their daily life, for example, banking fraud or identity theft. As a cardiologist and electro-physiologist, one of the authors (P. Ott, M.D.) has observed an increase in patients' awareness of security issues, who question the safety of implanted devices in the digital realm. We expect such concerns will become even more pressing. A small study has shown perceived security, safety, freedom from unwanted cultural and historical associations, and self-image must be taken into account when designing countermeasures for medical devices.⁵

We need more information about how concerned patients are about the security of the devices they are using. A user study could reveal what specific, additional steps patients are willing to take in order to increase security. This will give manufacturers valuable information. We will need to increase security awareness of all stakeholders, that is, manufacturers, patients, doctors, and medical institutions. Additionally, the devices' security states must be more visible, understandable, and accessible for all stakeholders.

IT infrastructure. In order to protect medical devices, the surrounding IT environment must be secured as well. Focusing on medical devices, we will refrain from enumerating regular countermeasures found in IT security. These are appropriate for health care security or medical device security as well, for example, erasing hard disks before disposing of them, backing up data, or BYOD (bring your own device) policies. Off-the-shelf devices like smartphones or tablets also increasingly store, process, and transmit sensitive medical data. This data must be protected from malware on these devices.

IT infrastructure must guarantee privacy of medical data according to the Health Insurance Portability and Accountability Act (HIPAA). However, safety is at stake as well. For medical devices, it is important to keep in mind regular IT devices pose a threat to medical devices also when they interoperate directly or indirectly. Most

importantly, medical devices should always assume their surroundings might have been compromised.

Conclusion

Securing medical devices means protecting human life, human health, and human well-being. It is also about protecting and securing the privacy of sensitive health information. We see an increase in the use of mobile medical applications as well as an increase in medical devices that use wireless communication and utilize Internet connections. New sensing technology provides opportunities for telemedicine with the promise to make health care more cost effective. Unless appropriate countermeasures are taken, the doors stand wide open for the misuse of sensitive medical data and even for malware and attacks that put human life in danger. 

References

1. Alemzadeh, H., Iyer, R.K. and Kalbarczyk, Z. Analysis of safety-critical computer failures in medical devices. *IEEE Security & Privacy* 11, 4, (July-Aug. 2013), 14–26.
2. *Boston Scientific*. PACEMAKER System Specification. 2007.
3. Denning, T., Fu, K. and Kohno, T. Absence makes the heart grow fonder: New directions for implantable medical device security. In *Proceedings of USENIX Workshop on Hot Topics in Security*, July 2008.
4. Denning, T., Matsuoka, Y. and Kohno, T. Neurosecurity: Security and privacy for neural devices. *Neurosurgical Focus* 27, 1 (July 2009).
5. Denning, T. et al. Patients, pacemakers, and implantable defibrillators: Human values and security for wireless implantable medical devices. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, 2010.
6. Food and Drug Administration. MAUDE—Manufacturer and User Facility Device Experience; <http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfMAUDE/search.cfm>
7. Food and Drug Administration. Is The Product A Medical Device? <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/ClassifyYourDevice/ucm051512.htm>
8. Food and Drug Administration. Medical Devices – Classify Your Medical Device; <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/ClassifyYourDevice/default.htm>
9. Food and Drug Administration Safety Communication: Cybersecurity for Medical Devices and Hospital Networks; June 2013. <http://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/ucm356423.htm>
10. Food and Drug Administration. Content of premarket submissions for management of cybersecurity in medical devices—Draft guidance for industry and Food and Drug administration staff, June 14, 2013; <http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/guidanceDocuments/ucm356186.htm>
11. Fox News. Antivirus Program Goes Berserk, Freezes PCs. Apr. 22, 2010.
12. Fu, K. and Blum, J. Controlling for cybersecurity risks of medical device software. *Commun. ACM* 56, 10 (Oct. 2013), 35–37.
13. Gollakota, S. et al. They can hear your heartbeats: Non-invasive security for implantable medical devices. In *Proceedings from SIGCOMM'11* (Toronto, Ontario, Canada, Aug. 15–19, 2011).
14. Halperin, D. et al. Security and privacy for implantable medical devices. *IEEE Pervasive Computing, Special Issue on Implantable Electronics*, (Jan. 2008).
15. Halperin, D. et al. Pacemakers and implantable cardiac

- defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 2008.
16. Hansen, J.A. and Hansen, N.M. A taxonomy of vulnerabilities in implantable medical devices. In *Proceedings of SPIMACS'10*, (Chicago, IL, Oct. 8, 2010).
17. Howard, M. and Lipner, S. *The Security Development Lifecycle*. Microsoft Press, 2006.
18. International Standards Organization. Medical devices—Application of risk management to medical devices. ISO 14971:2007.
19. Jee, E. et al. A safety-assured development approach for real-time software. *Proc. IEEE Int. Conf. Embed. Real-time Comput. Syst. Appl.* (Aug. 2010), 133–142.
20. Kaplan, D. Black Hat: Insulin pumps can be hacked. *SC Magazine*, (Aug. 04, 2011).
21. King, S.T. et al. Designing and implementing malicious hardware. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. Fabian Monrose, ed. USENIX Association, Berkeley, CA.
22. Kolata, G. Of fact, fiction and Cheney's defibrillator. *New York Times*, (Oct. 27, 2013).
23. Kramer, D.B. et al. Security and privacy qualities of medical devices: An analysis of fda postmarket surveillance. *PLoS ONE* 7, 7 (2012), e40200; doi:10.1371/journal.pone.0040200
24. Li, C., Raghunathan, A. and Jha, N.K. Improving the trustworthiness of medical device software with formal verification methods. *IEEE Embedded Systems Letters* 5, 3 (Sept. 2013), 50–53.
25. McGraw, G. Software security. *IEEE Security & Privacy* 2, 2 (Mar–Apr 2004), 80–83.
26. Nixon, C. et al. Academic Dual Chamber Pacemaker. University of Minnesota, 2008.
27. Ross, R.S. Guide for Conducting Risk Assessments. NIST Special Publication 800-30 Rev. 1, Sept. 2012.
28. Rostami, M., Juels, A. and Koushanfar F. Heart-to-Heart (H2H): Authentication for implanted medical devices. In *Proceedings for ACM SIGSAC Conference on Computer & Communications Security*. ACM, New York, NY, 1099–1112.
29. Sanger, D.E. and Shanker, T. N.S.A. devises radio pathway into computers. *New York Times* (Jan. 14, 2014).
30. Skorobogatov, S. and Woods, C. Breakthrough silicon scanning discovers backdoor in military chip, cryptographic hardware and embedded systems. *Lecture Notes in Computer Science* 7428 (2012), 23–40.
31. Sorber, J. et al. An amulet for trustworthy wearable mHealth. In *Proceedings of the 12th Workshop on Mobile Computing Systems & Applications*. ACM, New York, NY.
32. Venere, E. New firewall to safeguard against medical-device hacking. *Purdue University News Service*, Apr. 12, 2012.
33. Vockley, M. Safe and Secure? Healthcare in the cyberworld. AAMI (Advancing Safety in Medical Technology) BI&T – Biomedical Instrumentation & Technology, May/June 2012.
34. Weaver, C. Patients put at risk by computer viruses. *Wall Street Journal* (June 13, 2013).
35. Wei, S., Potkonjak, M. The undetectable and unprovable hardware Trojan horse. In *Proceedings of the ACM Design Automation Conference* (Austin, TX, May 29 – June 07, 2013).
36. Wirth, A. Cybercrimes pose growing threat to medical devices. *Biomed Instrum Technol.* 45, 1 (Jan/Feb 2011), 26–34.
37. World Health Organization. Medical device regulations: Global overview and guiding principles. 2003.

Johannes Sametinger (johannes.sametinger@jku.at) is an associate professor in the Department of Information Systems at the Johannes Kepler University Linz, Austria.

Jerzy Rozenblit (jr@ece.arizona.edu) is Distinguished Professor in the Department of Electrical and Computer Engineering/Dept. of Surgery at the University of Arizona, Tucson, AZ.

Roman Lysecky (rlysecky@ece.arizona.edu) is an associate professor in the Department of Electrical and Computer Engineering at the University of Arizona, Tucson, AZ.

Peter Ott (ottp@email.arizona.edu) is an associate professor in the College of Medicine, Sarver Heart Center at the University of Arizona, Tucson, AZ.

research highlights

P. 84

Technical Perspective **The Specialization Trend in Computer Hardware**

By Trevor Mudge

P. 85

Convolution Engine: Balancing Efficiency and Flexibility in Specialized Computing

By Wajahat Qadeer, Rehan Hameed, Ofer Shacham,
Preethi Venkatesan, Christos Kozyrakis, and Mark Horowitz

Technical Perspective

The Specialization Trend in Computer Hardware

By Trevor Mudge

THE BREAKDOWN OF Dennard scaling has become a new challenge that computer architects face as we move to smaller technology nodes. In the past, Dennard scaling meant that speed increased, power dropped, and area shrank. And, following Moore's Law, this triple win was achieved without appreciable increase in chip cost. This remarkable era in which Moore's Law held sway is drawing to a close for CMOS-based technology, and, as we transition to sub-20nm feature sizes, at least two of these improvements—speed and power—are grinding to a halt. In fact, several pundits have argued the 28nm node may be the cheapest node in terms of cost per transistor.

This slowdown in scaling will have a profound impact across the whole spectrum of computing, starting with the manner in which hardware systems are architected in a future where increased functionality and much tighter power constraints will be the norm. Nowhere is this more apparent than in mobile platforms, particularly smartphones, where there are ever-increasing demands for functionality without compromising battery life. It will no longer always be possible to simply move to the next-generation general-purpose computer to meet tighter energy-performance constraints.

An alternative that improves energy efficiency is specialization, but it only makes economic sense if there is significant demand. A balance can often be found by designing application-domain-specific components that have a degree of programmability, which is enough to open up significant application space. The work presented in the following paper does just that.

The authors present a programmable convolution engine that takes a 1D or 2D template (or stencil) and convolves it with a 1D or 2D array of data—a 2D image is a typi-

The authors present a detailed study to show where their design fits in the space between fixed-function hardware and general-purpose hardware.

cal example. The stencil is much smaller than the image. It can be as small as 3x3 pixels, whereas the image may have hundreds of millions of pixels. Its size is programmable, as is the function it performs. The applicability of this kind of operation is widespread, particularly in image processing, computer vision, and the emerging field of virtual reality. The rapid adoption of such functions into mobile platforms underscores the need for them to be performed in a power-efficient way: it is not difficult to justify a specialized processor for convolution. This notion was the impetus behind the authors' investigation into a programmable convolution engine.

Within the space of programmable convolution engines there are still important design choices to be made in order to make the most of the decisions to specialize. To guide these choices the authors make the crucial observation that "specialized units achieve most of their efficiency gains by tuning data storage structures to the data-flow and data-locality requirements of the algorithm." This

important observation greatly benefits power-performance efficiency. Convolution involves a great deal of data movement, for example, a large image must be accessed from memory. These accesses follow well-defined patterns that can be precomputed once the particular type of convolution and the size of the stencil are known. Thoughtful design allows these access patterns to be performed so as to minimize redundant accesses, greatly reducing power consumption while speeding the calculation.

To conclude, the authors present a detailed study to show where their design fits in the space between fixed-function hardware and general-purpose hardware. Specifically, they consider two ends of the spectrum: a custom implementation and a general-purpose processor augmented with SIMD instructions. The SIMD instructions model similar features found in commercial processors, such as Intel's Streaming SIMD Extensions and ARM's NEON extensions. They show their convolution engine implementation is 8–15 times more energy and area efficient than a general-purpose processor plus SIMD, but within a factor of 2–3 times of the custom unit. This analysis neatly illustrates the pros and cons of specialization with some degree of programmability. Finally, the analysis alone is worth study because it provides one of the clearest examinations of where energy and area are expended in today's processors. 

Trevor Mudge (tnm@eecs.umich.edu) is the Bredt Family Professor of Engineering in the EECS Department at the University of Michigan, Ann Arbor.

Copyright held by author.

Convolution Engine: Balancing Efficiency and Flexibility in Specialized Computing

By Wajahat Qadeer, Rehan Hameed, Ofer Shacham, Preethi Venkatesan, Christos Kozyrakis, and Mark Horowitz

Abstract

General-purpose processors, while tremendously versatile, pay a huge cost for their flexibility by wasting over 99% of the energy in programmability overheads. We observe that reducing this waste requires tuning data storage and compute structures and their connectivity to the data-flow and data-locality patterns in the algorithms. Hence, by backing off from full programmability and instead targeting key data-flow patterns used in a domain, we can create efficient engines that can be programmed and reused across a wide range of applications within that domain.

We present the Convolution Engine (CE)—a programmable processor specialized for the convolution-like data-flow prevalent in computational photography, computer vision, and video processing. The CE achieves energy efficiency by capturing data-reuse patterns, eliminating data transfer overheads, and enabling a large number of operations per memory access. We demonstrate that the CE is within a factor of 2–3× of the energy and area efficiency of custom units optimized for a single kernel. The CE improves energy and area efficiency by 8–15× over data-parallel Single Instruction Multiple Data (SIMD) engines for most image processing applications.

1. INTRODUCTION

Processors, whether they are the relatively simple RISC cores in embedded platforms, or the multibillion transistor CPU chips in Server/Desktop computers, are extremely versatile computing machines. They can handle virtually any type of workload ranging from web applications, personal spreadsheets, image processing workloads, and embedded control applications to database and financial applications. Moreover, they benefit from well-established programming abstractions and development tools, and decades of programming knowledge making it very easy to code new applications.

Processors, however, are also inefficient computing machines. The overheads of predicting, fetching, decoding, scheduling, and committing instructions account for most of the power consumption in a general-purpose processor core.^{2,7,16} As a result they often consume up to 1000× more energy than a specialized hardware block designed to perform just that particular task. These specialized hardware blocks also typically offer hundreds of times higher performance using a smaller silicon area. Despite these large inefficiencies, processors form the core of most computing systems owing to their versatility and reuse.

Over decades, we have been able to scale up the performance of general-purpose processors without using excessive power, thanks to advances in semiconductor device technology. Each new technology generation exponentially reduced the switching energy of a logic gate enabling us to create bigger and more complex designs with modest increases in power. In recent years, however, the energy scaling has slowed down,¹² thus we can no longer scale processor performance as we used to do. Today we fundamentally need to reduce energy waste if we want to scale performance at constant power.

This paper presents a novel highly efficient processor architecture for computational photography, image processing, and video processing applications, which we call the Convolution Engine (CE). With the proliferation of cheap high quality imagers, computational photography and computer vision applications are expected to be critical consumer computation workloads in coming years. Some example applications include annotated reality, gesture-based control, see-in-the-dark capability, and pulse measurement.

Many of these applications, however, will require multiple TeraOps/s of computation which is far beyond the capability of general processor cores especially mobile processors on a constrained power budget of less than 1 Watt. The three orders of magnitude advantage in compute efficiency of hardware accelerators, means that current mobile systems use heterogeneous computing chips combining processors and accelerators.^{11,15} An example accelerator is the video codec hardware employed in the mobile SOCs. However, these accelerators target either a single algorithm or small variations on an algorithm. Handling the diverse application set needed by these future smart devices requires a more programmable computing solution. Our CE provides that programmability while still keeping energy dissipation close to dedicated accelerators.

Our design approach is based on the observation that *specialized units achieve most of their efficiency gains by tuning data storage structures to the data-flow and data-locality requirements of the algorithm*. This tuning eliminates redundant data transfers and facilitates creation of closely coupled

The original version of this paper is entitled “Convolution Engine: Balancing Efficiency and Flexibility in Specialized Computing” and was published in *Proceedings of the 40th Annual International Symposium on Computer Architecture*, June 2013.

data-paths and storage structures allowing hundreds of low-energy operations to be performed for each instruction and data fetched. Processors can enjoy similar energy gains if they target computational motifs and data-flow patterns common to a wide-range of kernels in a domain. Our CE implements a generalized map-reduce abstraction, which describes a surprisingly large number of operations in image processing domain. The resulting design achieves up to two orders of magnitude lower energy consumption compared to a general-purpose processor and comes within 2–3× of dedicated hardware accelerators.

The next section provides an overview of why general processors consume so much power and the limitations of existing optimization strategies. Section 3 then introduces the convolution abstraction and the five application kernels we target in this study. Section 4 describes the CE architecture focusing primarily on features that improve energy efficiency and/or allow for flexibility and reuse. We then compare this CE to both general-purpose cores with SIMD extensions and to highly customized solutions for individual kernels in terms of energy and area efficiency. Section 5 shows that the CE is within a factor of 2–3× of custom units and almost 10× better than the SIMD solution for most applications.

2. BACKGROUND

The low efficiency of general-purpose processors is explained in Figure 1, which compares the energy dissipation of various arithmetic operations with the overall instruction energy for an extremely simple RISC processor. The energy dissipation of arithmetic operations that perform the useful work in a computation remains much lower than the energy wasted in the instruction overheads such as instruction fetch, decode, pipeline management, program sequencing, etc. The overhead is even worse for media processing applications which typically operate on short data requiring just 0.2–0.5 pJ (90 nm) of energy per operation, with the result that over 99% energy goes into overheads.

These overheads must be drastically reduced to increase energy efficiency. That places two constraints on the processor design, (i) the processor must execute hundreds of operations per instruction to sufficiently amortize instruction cost and (ii) the processor must also fetch little data, since even a cache hit costs 25 pJ (90 nm) per memory fetch, compared to 0.2–0.5 pJ for the arithmetic operations.

Figure 1. Comparison of functional unit energy with that of a typical RISC instruction in 90nm. Strategy for amortizing processor overheads includes executing hundreds of low-power operations per instruction.

RISC Instruction	Overhead	ALU	125 pJ
Load/Store	D-\$	Overhead	ALU
SP Floating Point		+	15–20 pJ
32-bit Addition		+	7 pJ
8-bit Addition		+	0.2–0.5 pJ

To get more than two orders of magnitude gain in efficiency

Overhead	+	+	+	hundreds	+	+	+
----------	---	---	---	----------	---	---	---

These two constraints seem contradictory as performing hundreds of operations per cycle would generally necessitate reading large amounts of data from the memory. These conditions could be reconciled, however, for algorithms where most instructions either operate on intermediate results produced by previous instructions, or reuse most of the input data used by the previous instructions. If an adequate storage structure is in place to retain this “past data” in the processor data-path, then large number of operations can be performed per instruction without needing frequent trips to the memory. Fortunately compute bound applications including most image processing and video processing algorithms are a good match for these constraints, providing large data-parallelism and data-reuse.

Most high-performance processors today already include SIMD units which are widely regarded as the most efficient general-purpose optimization for compute intensive applications. SIMD units typically achieve an order of magnitude energy reduction by simultaneously operating on many data operands in a single cycle (typically 8–16). However, as explained in Hameed et al.⁷ the resulting efficiency remains two orders of magnitude less than specialized hardware accelerators, as the SIMD model does not scale well to larger degrees of parallelism.

To better understand the architectural limitations of traditional SIMD units, consider the two-dimensional sum of absolute difference operation (SAD) operating on a 16-bit 8×8 block as shown in Listing 1.⁵ The 2D SAD operator is widely employed in multimedia applications such as H.264 video encoder to find the closest match for a 2D image sub-block in a reference image or video frame. Listing 1 carries out this search for every location in a *srchWinHeight* $\times$ *srchWinWidth* search window in the reference frame, resulting in four nested loops. All four loops are independent and can be simultaneously parallelized. At the same time, each SAD output substantially reuses the input data used to compute previous outputs, both in vertical and horizontal directions.

However, a typical SIMD unit with a register row size of 128 bits is only able to operate on elements that fit in one register row limiting the parallelism to the inner most loop. Trying to scale up the SIMD width to gain more parallelism requires either simultaneously reading multiple image rows from the register file (to parallelize across the 2nd most-inner loop), or simultaneously reading multiple overlapping rows of image data (to parallelize across multiple horizontal outputs). Neither support exists in the SIMD model.

Listing 1. 2D 8×8 sum of absolute difference operation (SAD), commonly employed in H.264 motion estimation.

```

for (sWinY = 0; sWinY < srchWinHeight; sWinY++) {
  for (sWinX = 0; sWinX < srchWinWidth; sWinX++) {
    sad = 0;
    for (y = 0; y < 8; y++) {
      for (x = 0; x < 8; x++) {
        cY = y + sWinY; cX = x + sWinX;
        sad += abs(ref[cY][cX] - cur[y][x]);
      }
    }
    outSad[sWinY][sWinX] = sad;
  }
}

```

Exploiting the data-reuse requires storing the complete 8×8 block in eight rows of the SIMD register file, so that the data is locally available for computing subsequent output pixels. This is straightforward in vertical direction, as every new output just needs one new row and the seven old rows could be reused. To get reuse in the horizontal direction, however, each of the eight rows must be shifted by one pixel before computing each new output pixel, incurring substantial instruction overhead. At the same time, since the shifting process destroys the old pixels, we are limited to getting reuse either in the vertical or horizontal direction but not both, with the result that each data item gets fetched eight times from the memory, wasting too much memory energy.

GPUs achieve a much higher degree of parallelism by using a large number of simple SIMD cores, each with local register resources, and very large memory bandwidth to maintain high computational throughput. While that results in great compute throughput, the energy consumption is also very high thanks to large data access cost. Measuring the performance and energy consumption of an optimized GPU implementation of H.264 SAD algorithm¹³ using GPUGPUSim simulator¹ with GPUWattch energy model,⁹ we ascertain that the GPU implementation runs 40× faster compared to an embedded 128-bit SIMD unit, but also consumes 30× higher energy. That further illustrates the need to minimize memory accesses and provide low-cost local data-accesses.

As the next section shows, the SAD operator belongs to a large class of algorithms which can be described as convolution-like and have the desired parallelism and reuse characteristics for efficient execution. Next section discusses this computational abstraction and provides a number of example applications.

3. COMPUTATIONAL MODEL

Equations (1) and (2) provide the definition of standard discrete 1D and 2D convolutions. When dealing with images, Img is a function from image location to pixel value, while f is the filter applied to the image. Practical kernels reduce computation (at a small cost of accuracy) by making the filter size small, typically in the order of 3×3 to 8×8 for 2D convolution:

$$(Img * f)[n] = \sum_{k=-\infty}^{\infty} Img[k] \cdot f[n-k] \quad (1)$$

$$(Img * f)[n, m] = \sum_{l=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} Img[k, l] \cdot f[n-k, m-l] \quad (2)$$

We generalize the concept of convolution by identifying two components of the convolution: a *map* operation and a *reduce* operation. In Equations (1) and (2), the map operation is the multiplication that is done on pairs of pixel and tap coefficient, and the reduce operation is the summation of all these pairs to the output value at location $[n, m]$. Replacing the map operation in Equation (2) from $x \cdot y$ to $|x - y|$ while leaving the reduce operation as summation, yields a sum of absolute numbers (SAD) function which is used for H.264's motion estimation. Further replacing the reduce operation from Σ to $\max$ will yield a max of absolute differences operation. Equation (3) expresses the computation model of our

CE, where f , *Map*, and *Reduce* (“ R ” in Equation 3) are the pseudo instructions, and c is the size of the convolution:

$$(Img * f)[n, m] \stackrel{\text{def}}{=} R_{|l| < c} \{ R_{|k| < c} \{ Map(Img[k], f[n-k, m-l]) \} \} \quad (3)$$

The convolution like data-flow works for many applications, but is sometimes limited by having a single associative operation in the reduction. There are a number of applications that have a data locality pattern similar to convolution, but need to combine results through a specific graph of operations rather than simple reduction. These applications can be handled by the CE by increasing the complexity of the *Reduce* operator to enable noncommutative functions, and allowing a different function to be used at each reduction stage. This generalized combining network extends the “reduction” stage to create a structure that can input a large number of values and then compute a small number of outputs through effectively a fused super instruction as shown in Figure 2.

The down side of this extension is that the placement of input into the combining tree is now significant; thus, to realize the full potential of the new generalized *reduce* operator, a high level of flexibility is required in the data supply network to move the needed data to the right position. This is achieved by extending the definition of the *map* operator to also support a data permutation network in addition to the already supported set of compute operators. These new enhancements to the *map* and *reduce* operators substantially boost their generalizability and applicability and thus increase the space of supported applications.

We now introduce a few kernels that we use in this paper and discuss how each of them maps into the computing abstractions we have defined above. Table 1 summarizes this information.

3.1. Motion estimation

Motion estimation is a key component of many video codecs including H.264, consuming about ~90% of the execution time for H.264 software implementations.^{4, 7} The kernel operates on subblocks of a video frame, trying to find each subblock's location in a previous and/or future reference frame of the video stream. In particular, in H.264, motion estimation is computed in two steps:

Integer Motion Estimation (IME): IME searches for the matching block in the reference image using the SAD

Figure 2. Generalized reduction unit fuses multiple operations into a super instruction.

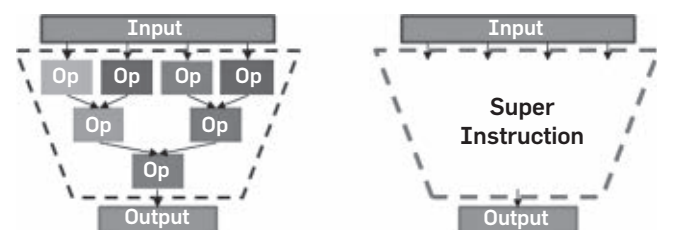


Table 1. Mapping kernels to convolution abstraction.

	Map	Reduce	Stencil sizes	Data flow
IME SAD	Abs diff	Add	4×4	2D convolution
FME 1/2 pixel up-sampling	Multiply	Add	6	1D horizontal and vertical convolution
FME 1/4 pixel up-sampling	Average	None	–	2D matrix operation
SIFT Gaussian blur	Multiply	Add	9, 13, 15	1D horizontal and vertical convolution
SIFT DoG	Subtract	None	–	2D matrix operation
SIFT extrema	Compare	Logical AND	9×3	2D convolution
Demosaic interpolation	Multiply	Complex	3	1D horizontal and vertical convolution

Some kernels such as subtraction operate on single pixels and thus have no stencil size defined. We call these as matrix operations. There is no reduce step for these operations.

operator previously described in Section 2. Note how SAD fits quite naturally to a CE abstraction: the *map* function is absolute difference and the *reduce* function is summation.

Fractional Motion Estimation (FME): FME refines the initial match obtained at the IME step to a quarter-pixel resolution. It first up-samples the block selected by IME, and then performs a slightly modified variant of the SAD operation. Up-sampling also fits nicely to the convolution abstraction and actually includes two convolution operations: first, the image block is up-sampled by two using a six-tap separable 2D filter. This part is purely convolution. Second, the resulting image is up-sampled by another factor of two by interpolating adjacent pixels, which can be defined as a *map* operator (to generate the new pixels) with no *reduce*.

3.2. SIFT

Scale Invariant Feature Transform (SIFT) looks for distinctive features in an image.¹⁰ To ensure scale invariance, Gaussian blurring and down-sampling is performed on the image to create a pyramid of images at coarser and coarser scales. A Difference-of-Gaussian (DoG) pyramid is then created by computing the difference between every two adjacent image scales. Features of interest are then found by looking at the scale-space extrema in the DoG pyramid.¹⁰

Gaussian blurring and down-sampling are naturally 2D convolution operations. Finding scale-space extrema is a 3D stencil computation, but we can convert it into a 2D stencil operation by interleaving rows from different images into a single buffer. The extrema operation is mapped to convolution using compare as a *map* operator and logical AND as the *reduce* operator.

3.3. Demosaic

Camera sensor output is typically a red, green, and blue (RGB) color mosaic laid out in Bayer pattern.³ At each location, the two missing color values are then interpolated using the luminance and color values in surrounding cells. Because the color information is undersampled, the interpolation is tricky; any linear approach yields color fringes. We use an implementation of Demosaic that is based upon adaptive color plane interpolation (ACPI),⁸ which computes image gradients and then uses a three-tap filter in the direction of smallest gradient. While this fits the generalize convolution flow, it requires a complex

“reduction” tree to implement the gradient-based selection. The data access pattern is also nontrivial since individual color values from the mosaic must be separated before performing interpolation.

4. CONVOLUTION ENGINE

Convolution operators are highly compute-intensive, particularly for large stencil sizes, and being data-parallel they lend themselves to vector processing. However, as explained earlier, existing SIMD units are limited in the extent to which they can exploit the inherent parallelism and locality of convolution due to the organization of their register files. The CE overcomes these limitations with the help of shift register structures. As shown in Figure 3 for the 2D convolution case, when such a storage structure is augmented with an ability to generate multiple shifted versions of the input data, it can fill 128 ALUs from just a small 16×8 2D register with low access energy as well as area. Similar gains are possible for 1D horizontal and 1D vertical convolutions. As we will see shortly, the CE facilitates further reductions in energy overheads by creating fused super-instructions introduced in Section 3.

The CE is developed as a domain specific hardware extension to Tensilica’s extensible RISC cores.⁶ The extension hardware is developed using Tensilica’s TIE language.¹⁴ The next sections discuss the key blocks in the CE extension hardware, depicted in Figure 4.

4.1. Register files

The 2D shift register is used for vertical and 2D convolution flows and supports vertical row shift: one new row of pixel data is shifted in as the 2D stencil moves vertically down into the image. The 2D shift register provides simultaneous access to all of its elements enabling the interface unit to feed any data element to the ALUs. 1D shift register is used to supply data for horizontal convolution flow. New image pixels are shifted horizontally into the 1D register as the 1D stencil moves over an image row.

The 2D Coefficient Register stores data that does not change as the stencil moves across the image. This can be filter coefficients, current image pixels in IME for performing SAD, or pixels at the center of Windowed Min/Max stencils. The results of convolution operations are either written back to the 2D Shift Register or the Output Register. A later

Figure 3. Implementation of 8×8 2D SAD operation that exploits parallelism in all four loops of Listing 1. The reference block resides in a 2D shift register while the current block is stored in a 2D register. Because both registers allow 2D access of the 8×8 block, 64 ALUs can operate in parallel. To enable an even larger degree of parallelism and to exploit data-reuse in the horizontal direction, the shift register generates pairs of multiple overlapping 8×8 blocks which are then fed to the ALU through a multiplexer. These pairs allow parallel execution of 128 ALUs generating two outputs in parallel. After the generation of four pairs of horizontal outputs, the shift register shifts up by one to make room for a new row of search window achieving vertical data-reuse.

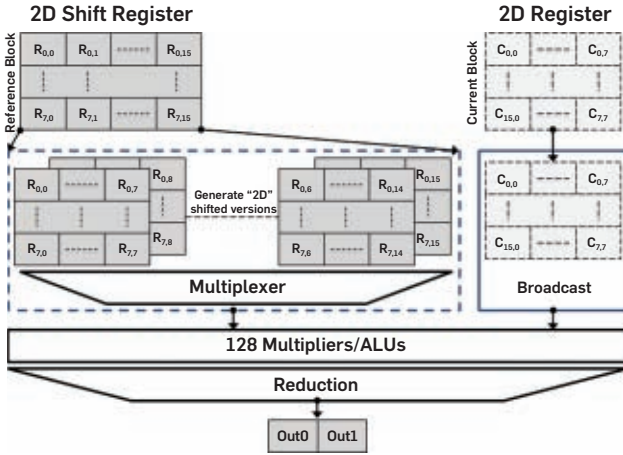
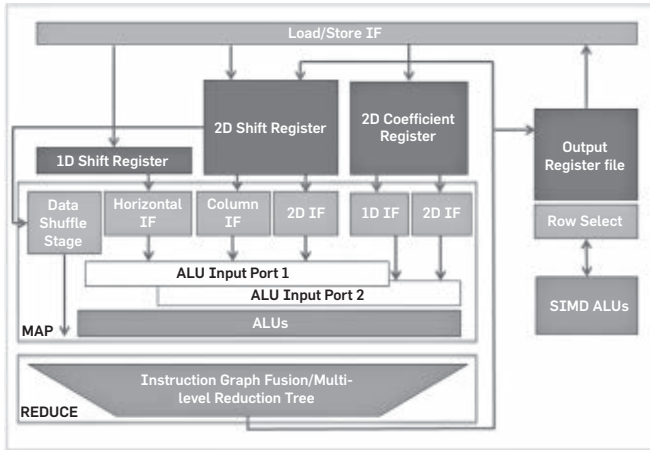


Figure 4. Block diagram of convolution engine. The interface units (IF) connect the register files to the functional units and provide shifted broadcast to facilitate convolution. Data shuffle (DS) stage combined with instruction graph fusion (IGF) stage create the generalized reduction unit, and is called the complex graph fusion unit.



section shows how the output register file also works as the vector register file for the vector unit shown in Figure 4.

4.2. Map and reduce logic

As described earlier we abstract convolution as a *map* and *reduce* step that transforms each input pixel into an output pixel. In our implementation, interface units and ALUs work together to implement the *map* operation; the interface units arrange the data as needed for the particular map

pattern and the functional units perform the arithmetic.

Interface units. The Interface Units (IF) arrange data from the register files into a specific pattern needed by the map operation. For 2D convolutions, multiple shifted 2D sub-blocks can be simultaneously accessed from the 2D register. Multiple block sizes such as 2×2 , 4×4 , 8×8 , etc. are supported and the appropriate size is selected based on convolution kernel size. Similarly for vertical convolution, multiple 2D register columns can be accessed in parallel, with support for multiple column access sizes. Finally, the 1D IF supports accessing multiple shifted 1D blocks from the 1D shift register for horizontal convolution. We are also exploring a more generalized permutation layer to support arbitrary maps.

Functional units. Since all data rearrangement is handled by the interface unit, the functional units are just an array of short fixed point two-input arithmetic ALUs. In addition to multipliers, we support absolute difference to facilitate SAD and other typical arithmetic operations such as addition, subtraction, and comparison. The output of the ALU is fed to the Reduce stage.

Reduce unit. The *reduce* part of the *map-reduce* operation is handled by a programmable reduce stage. Based upon the needs of our applications, we currently support arithmetic and logical reduction stages. The degree of reduction is dependent on the kernel size, for example a 4×4 2D kernel requires a 16 to 1 reduction whereas 8 to 1 reduction is needed for an 8-tap 1D kernel. Thus, the reduction stage is implemented as a combining tree and outputs can be tapped out from multiple stages of the tree.

To enable the creation of “super instructions” described in Section 3, we augment the combining tree to enable handle noncommutative operations by adding support for diverse arithmetic operations at different levels of the tree. This fusion increases the computational efficiency by reducing the number of required instructions and by eliminating temporary storage of intermediate data in register files. Because this more complex data combination need not be commutative, the right data (output of the map operation) must be placed on each input to the combining network. Thus, a “Data Shuffle Stage” is also added to the CE in the form of a very flexible swizzle network that provides permutations of the input data.

4.3. Other hardware

To facilitate vector operations on the convolution output, we have added a 32-element SIMD unit. This unit interfaces with the 2D Output Register and uses it as a Vector Register file. This unit is wider than typical SIMD units, as it operates on intermediate data generated by convolution data path and thus is not constrained by data memory accesses. Despite being wider, the vector unit is still lightweight as it only supports basic vector add and subtract type operations and has no support for higher cost operations such as multiplications.

Because an application may perform computation that conforms neither to the convolution block nor to the vector unit, or may otherwise benefit from a fixed function implementation. If the designer wishes to build a customized

unit for such computation, the CE allows the fixed function block access to its Output Register File. This model is similar to a GPU where custom blocks are employed for rasterization and such, and that work alongside the shader cores. For these applications, we created three custom functional blocks to compute motion vector costs in IME and FME and the Hadamard Transform in FME.

4.4. Resource sizing

Energy efficiency considerations and resource requirements of target applications drive the sizes of various resources within CE. As shown in Hameed et al.,⁷ amortizing the instruction cost requires performing hundreds of ALU operations per instruction for media processing applications based on short 8-bit Addition/Subtraction operations. Many convolution flow applications are, however, based on higher energy multiplication operations. Our analysis shows that for multiplication-based algorithms, 50–100 operations per instruction are enough to provide sufficient amortization. Increasing the number of ALUs much further than that gives diminishing returns and increases the size of storage required to keep these units busy, thus increasing storage area and data-access energy. Thus for this study we choose an ALU array size of 128 ALUs, and size the rest of the resources accordingly to keep these ALUs busy. To provide further flexibility we allow powering off half of the ALU array and compute structures. The size and capability of each resource is presented in Table 2. These resources support filter sizes of 4, 8, and 16 for 1D filtering and 4×4 , 8×8 , and 16×16 for 2D filtering. Notice that the register file sizes deviate from power of 2 to efficiently handle boundary conditions common in convolution operations.

4.5. Programming the convolution engine

CE is implemented as a processor extension and adds a small set of instructions to the processor ISA. These CE instructions can be issued as needed in regular C code through compiler intrinsics. Table 3 lists the major CE instructions. Configuration instructions set kernel parameters such as convolution size, ALU operation to use at Map and Reduction stages, etc. Then, there are load and store operations for each register resource. Finally, there are the compute instructions, one for each of the three supported convolution flows—1D horizontal, 1D vertical, and 2D. For example the CONVOLVE_2D instruction reads one set

Table 2. Sizes for various resources in CE.

	Resource sizes
ALUs	128 × 10 bit ALUs
1D shift register	80 × 10 bit
2D input shift register	16 rows × 36 cols × 10 bit
2D output shift register	16 rows × 36 cols × 10 bit
2D coefficient register	16 rows × 16 cols × 10 bit
Horizontal interface	4, 8, 16 kernel patterns
Vertical interface	4, 8, 16 kernel patterns
2D interface	4 × 4, 8 × 8, and 16 × 16 patterns
Reduction tree	4:1, 8:1, ..., 128:1

of values from 2D and coefficient registers, performs the convolution and write the result into the row 0 of 2D output register.

The code example in Listing 2 brings it all together and implements a $2D \ 8 \times 8$ Filter. First the CE is set to perform multiplication at MAP stage and addition at reduce stage which are the required setting for filtering. Then the convolution size is set which controls the pattern in which data is fed from the registers to the ALUs. Filter tap coefficients

Table 3. Major instructions added to processor ISA.

	Description
SET_CE_OPS	Set arithmetic functions for MAP and REDUCE steps
SET_CE_OPSIZE	Set convolution size
LD_COEFF_REG_n	Load n bits to specified row of 2D coeff register
LD_1D_REG_n	Load n bits to 1D shift register. Optional Shift left
LD_2D_REG_n	Load n bits to top row of 2D shift register. Optional shift row down
ST_OUT_REG_n	Store top row of 2D output register to memory
CONVOLVE_1D_HOR	1D convolution step—input from 1D shift register
CONVOLVE_1D_VER	1D convolution step—column access to 2D shift register
CONVOLVE_2D	2D convolution step with 2D access to 2D shift register

Listing 2. Example C code implements 8×8 2D filter for a vertical image stripe and adds 2 to each output.

```
// Set MAP function = MULT, Reduce function = ADD
SET_CE_OPS (CE_MULT, CE_ADD);

// Set convolution size 8
SET_CE_OPSIZE(8);

// Load eight rows of eight 8-bit coefficients
// into Coeff Reg's rows 0 to 7
for(i = 0; i < 8; i++){
    LD_COEFF_REG_128(coeffPtr, 0);
    coeffPtr += coeffWidth;
}

// Load & shift seven rows of sixteen input pixels
// into 2D shift register
for(i = 0; i < 7; i++){
    LD_2D_REG_128(inPtr, SHIFT_ENABLED);
    inPtr += width;
}

// Filtering loop
for (y = 0; y < height; y++) {
    // Load & Shift 16 more pixels
    LD_2D_REG_128(inPtr, SHIFT_ENABLED);

    // Filter first 8 locations. Because we have
    // access to 128-ALUS, we can filter two 8x8
    // blocks in parallel
    for(RW_OFFSET = 0; RW_OFFSET < 8; RW_OFFSET+=2){
        CONVOLVE_2D(RW_OFFSET, RW_OFFSET);
    }

    // Add 2 to row 0 of output register
    SIMD_ADD_CONST (0, 2);

    // Store 8 output pixels
    ST_OUT_REG_64(outPtr);

    inPtr += width;
    outPtr += width;
}
```

It is important to note that unlike a stand-alone accelerator the sequence of operations in CE is completely controlled by the C code which gives complete flexibility over the algorithm. For example, in the filtering code above, it is possible for the algorithm to produce one CE output to memory and then perform a number of non-CE operations on that output before invoking CE to produce another output.

To evaluate the efficiency of the CE, we map each target application described in Section 3 on a chip multiprocessor (CMP) comprised of two CEs. To quantify the performance and energy cost of such a programmable unit, we also built

The diagram illustrates the high-level architecture of the proposed GPU. It is divided into two main sections: **MAP** and **REDUCE**.

MAP Stage:

- Input:** 256-bit data from the **Load/Store IF** block.
- Parallel Processing:**
 - Path 1:** 256-bit data is processed by a **1D Shift Register** (40 x 10-bit) and then a **Data Shuffle Stage**.
 - Path 2:** 256-bit data is processed by an **8 Pixel Rows** block, then a **2D Shift Register** (16 x 18 x 10-bit), and finally a **Column IF** block.
 - Path 3:** 256-bit data is processed by an **8 x 8 Coeff Block**, then a **2D Coefficient Register** (16 x 16 x 10-bit), and finally a **1D IF** block.
- Reduction:** The outputs from the three paths are fed into **ALU input Port 1** and **ALU input Port 2** (both 128 x 10-bit). These feed into **ALUs** (128 x 10-bit Multiplies), which output 64 x 20-bit data.

REDUCE Stage:

- Input:** 64 x 20-bit data from the MAP stage.
- Processing:** The data is processed by an **Instruction Graph Fusion/ Multi-level Reduction Tree** (Up to 128:1 Reduce).
- Output:** The output is fed into a **Filtered Data Output Register file** (16 x 18 x 10-bit), which also receives **Row Select** (160-bit) and **SIMD ALUs** (16 x 10-bit) for further processing.

To quickly simulate and evaluate the CMP configurations, we created a multiprocessor simulation framework that employs Tensilica’s Xtensa Modeling Platform (XTMP) to perform cycle accurate simulation of the processors and caches. For energy estimation we use Tensilica’s energy explorer tool, which uses a program execution trace to give a detailed analysis of energy consumption in the processor core as well as the memory system. The estimated energy consumption is within 30% of actual energy dissipation. To account for interconnection energy, we created a floor plan for the CMP and estimated the wire energies from that. That

A bar chart comparing the energy consumption of three different engines (Custom, Convolution Engine, and SIMD) across five different processing tasks (SIFT-DoG, SIFT-Extrema, H.264-FME, H.264-IME, and Demosaic). The y-axis represents 'Energy normalized to custom (lower is better)' on a logarithmic scale from 0 to 100. The x-axis lists the tasks. For each task, there are three bars: a dark gray bar for the Custom engine (always at 1.0), a light gray bar for the Convolution Engine, and a medium gray bar for the SIMD engine. The SIMD engine consistently shows the highest energy consumption across all tasks, while the Convolution Engine shows intermediate values.

Task	Custom	Convolution Engine	SIMD
SIFT-DoG	1.0	~2.0	~15.0
SIFT-Extrema	1.0	~3.0	~50.0
H.264-FME	1.0	~2.5	~12.0
H.264-IME	1.0	~2.5	~25.0
Demosaic	1.0	~2.5	10.0

Ops/mm² normalized to custom
(higher is better)

Operation	Custom	Convolution Engine	SIMD
SIFT-DoG	1.0	~0.3	~0.06
SIFT-Extrema	1.0	~0.15	~0.03
H.264-FME	1.0	~0.3	~0.05
H.264-IME	1.0	~0.35	~0.06
Demosaic	1.0	~0.15	~0.08

interconnection energy was then added to energy estimates from Tensilica tools. The simulation results employ 90 nm technology at 1.1 V operating voltage with a target frequency of 450 MHz. All units are pipelined appropriately to achieve the frequency target.

Figures 6 and 7 compare the performance and energy dissipation of the proposed CE against a 128-bit data-parallel (SIMD) engine and the custom accelerator implementation for each of the five algorithms of interest. In most cases we used the SIMD engine as a 16-way 8-bit data-path, but in a few examples we created 8-way 16-bit data-paths. For our algorithms, making this unit wider did not change the energy efficiency appreciably.

The fixed function data points truly highlight the power of customization: for each application a customized accelerator requires 8–50 $\times$ less energy compared to SIMD engine. Performance per unit area is also 8–30 $\times$ higher than the SIMD implementation. Demosaic achieves the smallest improvement (8 $\times$) because it generates two new pixel values for every pixel that it loads from the memory. Therefore, after the customization of compute operations, loads/stores and address manipulation operations become the bottleneck and account for approximately 70% of the total instructions.

Note the biggest gains were in IME and SIFT extrema calculations. Both kernels rely on short integer *add/subtract* operations that are very low energy (relative to the *multiply* used in filtering and up-sampling). As previously discussed, for SIMD implementation the instruction overheads and data access energy are still large relative to the amount of computation done. Custom accelerators, on the other hand, are able to exploit the parallelism and data-reuse in their respective kernels, fully amortizing instruction and data fetch.

We can now better understand where the CE stands. The architecture of the CE is closely matched to the data-flow of convolution-based algorithms, therefore the instruction stream difference between fixed function units and the CE is very small. Compared to a SIMD implementation, the CE requires 8–15 $\times$ less energy with the exception of Demosaic that shows an improvement of 4 $\times$ while the performance to area ratio of CE is 5–6 $\times$ better. Again Demosaic is at the low end of the gain as a consequence of the abundance of loads and stores. If we discount the effect of memory operations from Demosaic, assuming its output is pipelined into another convolution like stage in the image pipeline, the CE-based Demosaic is approximately 7 $\times$ better than SIMD and within 6 $\times$ of custom accelerator. The higher energy ratio compared to a custom implementation points up the costs of the more flexible communication in the generalized reduction.

However, for all the other applications the energy overhead of the CE compared to fixed function units stands at a modest 2–3 $\times$, while the area overhead is just 2 $\times$. While these overheads are small, to better understand the sources of these inefficiencies, Figures 8 and 9 create three different implementations of each application, moving from a custom implementation, to one with flexible registers, but fixed computation, and to the fully flexible CE.

Figure 8. Change in energy consumption as programmability is incrementally added to the core.

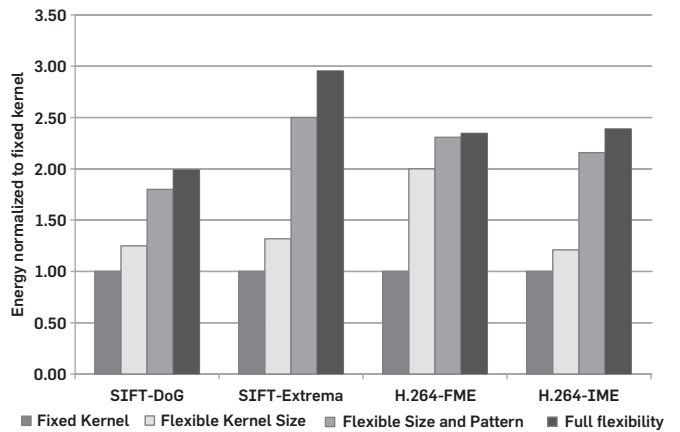
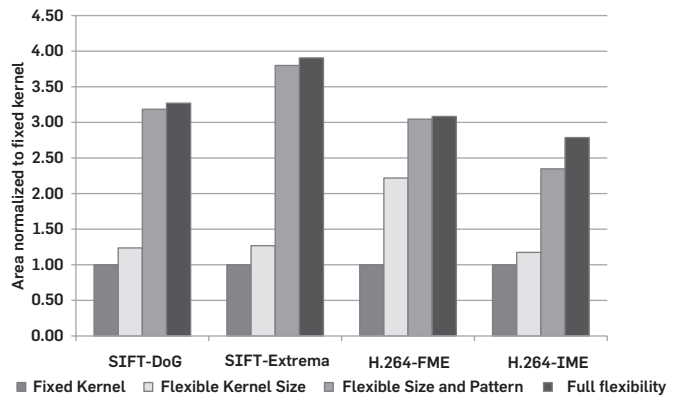


Figure 9. Increase in area as programmability is incrementally added to the core.



The figures show that the biggest impact on energy efficiency takes place when the needed communication paths become more complex. This overhead is more serious when the fundamental computation energy is small. In general, the communication path complexity grows with the size of the storage structures, so over-provisioning registers as is often needed in a programmable unit hurts efficiency. This energy overhead is made worse since such structures not only require more logic in terms of routing and muxing but also have a direct impact on the leakage energy. On the other hand, more flexible functional units have small overheads, which provides flexibility at low cost.

6. CONCLUSION

As specialization emerges as the main approach to addressing the energy limitations of current architectures, there is a strong desire to make maximal use of these specialized engines. This in turn argues for making them more flexible, and user accessible. While flexible specialized engines might sound like an oxymoron, we have found that focusing on the key data-flow and data-locality patterns within broad domains allows one to build a highly energy efficient engine, that is, still user programmable. We presented the

CE which supports a number of different algorithms from computational photography, image processing and video processing, all based on convolution-like patterns. A single CE design supports applications with convolutions of various size, dimensions, and type of computation. To achieve energy efficiency, CE captures data-reuse patterns, eliminates data transfer overheads, and enables a large number of operations per cycle. CE is within a factor of 2–3× of the energy and area efficiency of single-kernel accelerators and still provides an improvement of 8–15× over general-purpose cores with SIMD extensions for most applications. While the CE is a single example, we hope that similar studies in other application domains will lead to other efficient, programmable, and specialized accelerators.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency under Contract No. HR0011-11-C-0007. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency. 

References

- Bakhoda, A., Yuan, G., Fung, W.W.L., Wong, H., Aamodt, T.M. Analyzing CUDA workloads using a detailed GPU simulator. In *ISPASS: IEEE International Symposium on Performance Analysis of Systems and Software* (2009).
- Balfour, J., Dally, W., Black-Schaffer, D., Parikh, V., Park, J. An energy-efficient processor architecture for embedded systems. *Comput. Architect. Lett.* 7, 1 (2007), 29–32.
- Bayer, B. *Color Imaging Array*. US Patent Application No. 3971065 (1976).
- Chen, T.-C., Chien, S.-Y., Huang, Y.-W., Tsai, C.-H., Chen, C.-Y., Chen, T.-W., Chen, L.-G. Analysis and architecture design of an HDTV720p 30 frames/sec H.264/AVC encoder. *IEEE Trans. Circuits Syst. Video Technol.* 16, 6 (2006), 673–688.
- Corbal, J., Valero, M., Espasa, R. Exploiting a new level of DLP in multimedia applications. In *Proceedings of the 32nd Annual International Symposium on Microarchitecture* (Nov. 1999), 72–79.
- Gonzalez, R. Xtensa: A configurable and extensible processor. *Micro IEEE* 20, 2 (Mar. 2000), 60–70.
- Hameed, R., Qadeer, W., Wachs, M., Azizi, O., Solomatnikov, A., Lee, B.C., Richardson, S., Kozyrakis, C., Horowitz, M. Understanding sources of inefficiency in general-purpose chips. In *ISCA '10: Proceedings of the 37th Annual International Symposium on Computer Architecture* (2010), ACM.
- Hamilton, J.F., Adams, J.E. *Adaptive Color Plane Interpolation in Single Sensor Color Electronic Camera*. US Patent Application No. 5629734 (1997).
- Leng, J., Gilani, S., Hetherington, T., Tantawy, A.E., Kim, N.S., Aamodt, T.M., Reddi, V.J. GPUWattch: Enabling energy optimizations in GPGPUs. In *ISCA 2013: International Symposium on Computer Architecture* (2013).
- Lowe, D. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision* 60, 2 (2004), 91–110.
- NVIDIA Inc. Tegra mobile processors. <http://www.nvidia.com/object/tegra-4-processor.html>.
- Shacham, O., Azizi, O., Wachs, M., Qadeer, W., Asgar, Z., Kelley, K., Stevenson, J., Solomatnikov, A., Firoozshahian, A., Lee, B., Richardson, S., Horowitz, M. Rethinking digital design: Why design must change. *IEEE Micro* 30, 6 (Nov. 2010), 9–24.
- Stratton, J.A., Rodrigues, C., Sung, I.-J., Obeid, N., Chang, L.W., Anssari, N., Liu, G.D., Hwu, W.-M.W. Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing. IMPACT Technical Report. In *IMPACT-12-01*, 2012.
- Tensilica Inc. Tensilica Instruction Extension (TIE) Language Reference Manual.
- Texas Instruments Inc. OMAP 5 platform. www.ti.com/omap.
- Venkatesh, G., Sampson, J., Goulding, N., Garcia, S., Bryksin, V., Lugo-Martinez, J., Swanson, S., Taylor, M.B. Conservation cores: Reducing the energy of mature computations. In *ASPLOS'10* (2010), ACM.

Wajahat Qadeer and Rehan Hameed ([wqadeer, rhameed]@gmail.com), Palo Alto, CA.

Ofer Shacham (shacham@alumni.stanford.edu), Google, Mountain View, CA.

Preethi Venkatesan (preethiv@stanford.edu), Intel Corporation, Santa Clara, CA.

Christos Kozyrakis and Mark Horowitz ([kozyraki, horowitz]@stanford.edu), Stanford University, Stanford, CA.

©ACM 0001-0782/14/1100 \$15.00.

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

Rutgers, The State University of New Jersey Assistant Professor

The Department of Management Science and Information Systems of Rutgers Business School-Newark and New Brunswick invites applications for a tenure-track position at the Assistant Professor rank to start in September 2015.

This position is focused in the area of information systems and the candidate must be an active researcher and have strong record of scholarly excellence. Special consideration will be given to candidates with knowledge in any of the following areas: data mining, machine learning, security, data management and analytical methods related to business operations.

A letter of application articulating the candidate's fit (in terms of research and teaching) with the position description, a curriculum vitae, and the names and contact information of three persons that can provide references should be sent electronically to Luz Kosar at: kosar@business.rutgers.edu

Luz Kosar,
MSIS
Rutgers Business School –
Newark and New Brunswick
1 Washington Park #1068
Newark, New Jersey 07102-1895

University of Central Missouri Assistant Professor of Computer Science

The Department of Mathematics and Computer Science at the University of Central Missouri is accepting applications for four tenure-track and several non-tenure track positions in Computer Science beginning August 2015 at the rank of Assistant Professor. The UCM Computer Science program has 18 full time faculty and offers undergraduate and master programs in Computer Science. The department is expecting to launch a cybersecurity program in Fall 2015. We are looking for faculty excited by the prospect of shaping our department's future and contributing to its sustained excellence.

Tenure Track Positions (#997516 and #997517): Ph.D. in Computer Science by August 2015 is required. All areas in computer science will be considered with preference given to candidates with expertise in Big Data Analytics, Cybersecurity, Machine Learning or Software Engineering.

Non-Tenure Track Positions (#997495): Ph.D. in Computer Science or a closely related area is preferred. ABD will be considered. Previous college/university teaching experience is highly desirable.

To apply online, go to <https://jobs.ucmo.edu>. Apply to positions #997516, #997517 or #997495. Initial screening of applications begins March 1, 2015, and continues until positions are filled. Contact: Dr. Songlin Tian. Email: tian@ucmo.edu. Phone: 660-543-4930. Fax: 660-543-8013



TENURE-TRACK AND TENURED POSITIONS IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

The newly launched **ShanghaiTech University** invites highly qualified candidates to fill **multiple 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 built as a world-class research university for training future generations of scientists, entrepreneurs, and technological leaders. Located in Zhangjiang High-Tech Park in the cosmopolitan Shanghai, ShanghaiTech is ready to trail-blaze a new education system in China. Besides establishing and maintaining a world-class research profile, faculty candidates are also expected to contribute substantially to graduate and undergraduate education within the school.

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, visit <http://www.shanghaitech.edu.cn>.

Deadline: April 30, 2015



Faculty Positions in the Institute for Advanced Computational Science

Applications are invited for four tenure-track faculty positions of any rank (including endowed chairs), in applied mathematics and computer science in the Institute for Advanced Computational Science (IACS) at Stony Brook University. Candidates wishing to apply should have a doctoral degree in Applied Mathematics or Computer Science, though a degree in related fields may be considered. Ten years of faculty or professional experience is required for a senior position along with a demonstrated record of publications and research funding. A demonstrated record of publications and a demonstrated potential for research funding is required for any junior faculty. The selected candidate is expected to participate in interdisciplinary program development within the Institute and to establish a research program with a solid funding base through both internal and external collaborations. Of specific interest is research in, for example, programming models, algorithms, or numerical representations that advance scientific productivity or broaden the benefit and impact of high-performance computing. The selected candidates will have access to world-class facilities including those at nearby Brookhaven National Laboratory.

The Institute for Advanced Computational Science (<http://iacs.stonybrook.edu/>) was established in 2012 with an endowment of \$20M, including \$10M from the Simons Foundation. The current ten faculty members will double in number over the next few years to span all aspects of computation with the intent of creating a vibrant multi-disciplinary program. IACS seeks to make sustained advances in the fundamental techniques of computation and in high-impact applications including engineering and the physical, life, and social sciences. Our integrated, multidisciplinary team of faculty, students, and staff overcome the limitations at the very core of how we compute, collectively take on challenges of otherwise overwhelming complexity and scale, and individually and jointly define new frontiers and opportunities for discovery through computation. In coordination with the Center for Scientific Computing at Brookhaven National Laboratory, our dynamic and diverse institute serves as an ideal training and proving ground for new generations of students and researchers, and provides computational leadership and resources across the SBU campus and State of New York.

The search will remain open until suitable candidates are found with the first round of applications due May 15, 2015. All candidates must submit the required documentation online through the link provided below. Please input a cover letter, your curriculum vitae, a research plan (max. 2 pages) which should also describe how graduate and undergraduate students participate, a one-page statement of your teaching philosophy, a publication list, your funding record, and three reference letters to: <https://iacs-hiring.cs.stonybrook.edu>.

Stony Brook University/SUNY is an equal opportunity, affirmative action employer.

[CONTINUED FROM P. 96] 2×10^{30} Joules of energy—astronomically serious production—and money—for any power utility.

But there's an even greater opportunity in those numbers. How much energy would be required to actually blow apart a planet like Mercury? Simple physics shows the planet has a gravitational binding energy of just 2×10^{30} J. If that much energy is deposited into it, the planet will explode, like Alderaan in the original *Star Wars* movie.

Now I'm sure you'll agree that mining a planet would probably take time, even with today's best robot-army cascade. But once we've extracted the ores the first time, I'm about to show you a method to initiate exponential growth in yields. Your robot army can use the tremendous energy of solar power to lift the ores with space elevators, up to where the robots can construct solar sails to carry it out of the inner solar system. When they reach icy bodies like Saturn's moons, the robots can get hydrogen and deuterium there, and build fission-fusion-propelled rocket engines. With fusion, those engines will achieve a multiplier of 5 to 10 in available power. That's a big gain factor, and the investment already yields a big payoff if you stop there.

But there's a way to achieve even more from this seed investment. Suppose we send our swarm of robot fusion-fission-powered rockets to an exoplanet in another star system. By accelerating these rockets, the energy we obtain can be transferred into the kinetic energy in the swarm of rockets. When they arrive at the exoplanet, they can blow it up just by colliding with it.

Some of our robots can slow down, replicate near the exoplanet debris, and mine the fissionable ores that have been dispersed. And now the process can be started again for another iteration, and another, and another. Each would yield enough energy that we could split the swarm in two and target two more exoplanets in another round. At each iteration, we'll be able to skim off a tiny fraction of the energy yield, return it to us back home, and enjoy wealth beyond previous human dreams.

Congratulations. You are now in the starship-construction industry.

The robots will have such numbers and power that their fabricators will be able to produce palatial starships for us all, with luxuries like this fabulous hotel.

It will be enormously profitable to the investor who is patient, and continue into the indefinite future. Our robots can perform as “universal constructors,” or self-replicating automata like those devised by the great Hungarian-American mathematician and physicist John von Neumann. He proved long ago that the most effective way to mine planets is to use the exponential growth of “von Neumann machines” in his masterpiece *Theory of Self-Reproducing Automata*.

Yes, young man in the front row, I agree it would be unethical to destroy inhabited planets. We can program the robot rocket swarm to survey its target region and bypass planets with biospheres. That's one reason to choose baked, lifeless targets like Mercury.

Yes, lady in the front row, the programming of these robot rocket swarms must be reliable and robust. I know cosmic rays perturb processor programs by changing bits in the memory systems when they penetrate. There's nothing we can do but use sophisticated error-correcting encoding schemes like Reed-Solomon encoding, which prevents loss of information even if some of the data is erroneous. We wouldn't want the swarm of robots to forget their safety instructions and target Earth or other solar system planets. There's quite enough debris in our solar system without creating more asteroids that might cross Earth's orbit.

Speaking of debris, we have observed debris rings encircling stars like Beta Pictoris, the second brightest star in the constellation Pictor, for decades. Usually they are interpreted as planet formation, but what if other, alien, investors have already started exploiting the wealth-generating exponential cascade of robot swarms? The resources are clearly there to exploit. Maybe we should send robot rockets to some of those debris rings to mine them. The opportunities are clearly there for the wise and patient investor. The robots will have such numbers and power that their fabricators will be able to produce palatial starships for us all, with luxuries like this fabulous hotel.

Yes, I know fusion-fission rockets would require decades to reach exoplanets. The time to full return on investment may be long, but we can take out some profits before the swarm leaves the solar system without compromising the overall plan. Still, the returns of an indefinite revenue stream are beyond measure.

Sir, far in the back, I understand your concern that releasing swarms of planet-destroying robots into the galaxy might threaten sentient life-forms or provoke a reaction if there are interstellar security forces. But after decades of SETI searches, no signals from intelligent beings have been detected, nor have signs of interstellar policing been observed. Until evidence of such things emerges, I count the potential profits of planet mining as far outweighing any ethical reservations.

I see smiling faces, so thank you for your attention. I have just one parting request. When you build the first pioneering swarm, take me along as a partner to report growth progress and track profits. I want to see the fireworks. 

David Allen Batchelor (batchelor@alum.mit.edu) is a computer engineer for data systems on the Earth Science Data and Information System (ESDIS) Project at NASA Goddard Space Flight Center, Greenbelt, MD. His first science fiction novel was *The Metalmark Contract* published in 2011 by Black Rose Writing, Castroville, TX.

From the intersection of computational science and technological speculation,
with boundaries limited only by our ability to imagine what could be.

DOI:10.1145/2737915

David Allen Batchelor

Future Tense

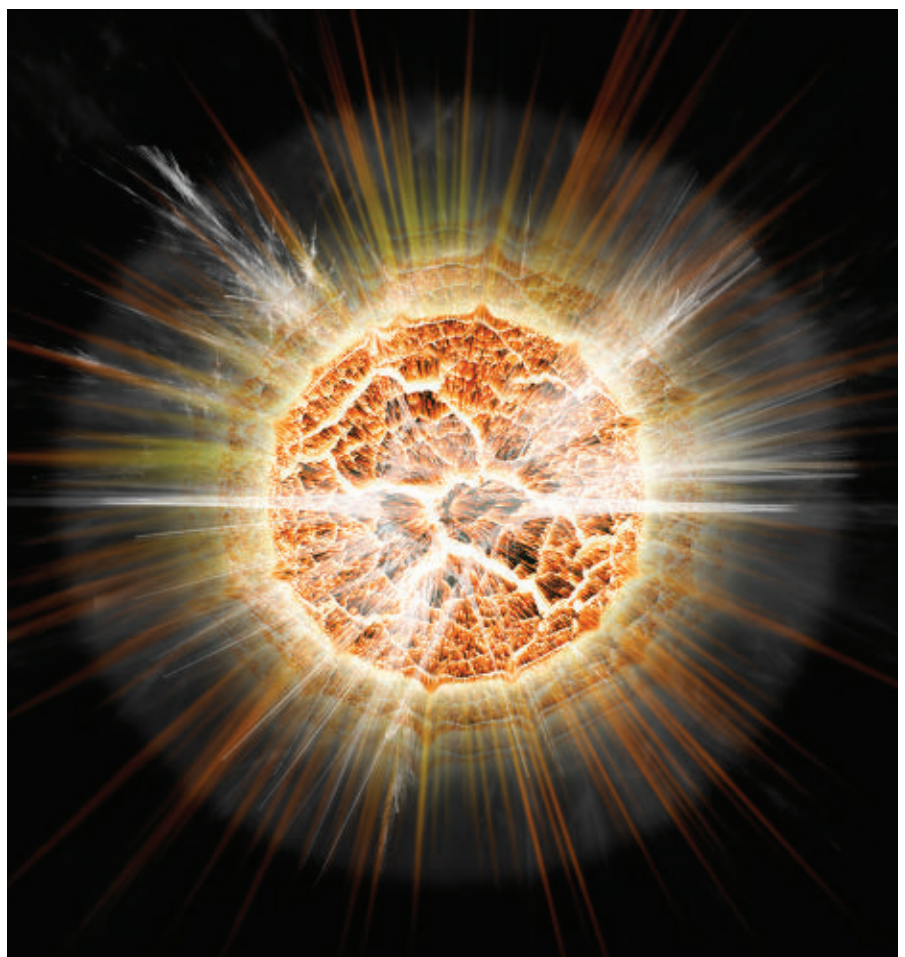
The Wealth of Planets

*Launch swarms of self-replicating robots
to exploit the most lucrative of resources.*

WELCOME, FELLOW WEALTH-MAKERS, to our seminar on how to initiate and maintain exponential growth of our own wealth-generating technology system. The ballroom here at the Lunar Hotel Armani Mare Serenitatis is a fitting location, with its opulent furnishings, luxurious lunar accommodations, and spectacular view of Earth. But the wealth-generating system I am about to describe will make you remember this hotel as if it were a mud hovel when you look back on it from your future dream starship.

Now that we have a commercial lunar economy, the resources of the solar system await those prepared to boldly commit the seed investment in the next big growth opportunity. I know venture capital is still flowing into lunar development, but this seminar is about an opportunity orders of magnitude greater—the resources of planet Mercury. Located two-and-a-half times closer to the Sun than our Earth, it features six-and-a-half times as much solar energy for free, every day, 24/7 ... or rather 1,400/ ... whatever a week would be on Mercury. Anyway, there's constant sunlight six times more powerful than the lunar photovoltaic systems will ever produce. And the opportunity to be first to utilize all that power is just the low-hanging fruit for wealth-makers like us to pick.

Our lunar complexes were built by commercially available robot fabricator cascades. You've all seen their work, powered by free solar energy, robots building robots building robots until the work force is large enough



for construction projects. But what would happen if we installed such a robot army on Mercury? I know, it's not a hospitable environment for hotels like this or for space tourism ... Yet. But that makes it all the more attractive for this reason: A robot army can mine the planet without restrictions like the lunar zoning regulations that just passed here.

An ambitious investor, or consortium of investors, could use available robot-army technology to extract all fissionable minerals—uranium, thorium, even radioactive potassium—and obtain fuel for the greatest growth investment ever. There's a total of 10^{16} kg of U and Th in the planet. Nuclear breeder reactions with that fuel would yield more than [CONTINUED ON P. 95]

6th Annual ACM SIGPLAN Conference on

Systems,
Programming,
Languages, and
Applications:
Software for
Humanity

Location

Sheraton Pittsburgh at
Station Square Hotel

Events

- 30th Annual OOPSLA
- Onward!
- Wavefront
- Generative Programming and Component Engineering (GPCE)
- Dynamic Languages Symposium (DLS)
- Software Language Engineering (SLE)
- Pattern Languages of Programming (PLoP)
- and more

General Chair

Jonathan Aldrich
Carnegie Mellon University

OOPSLA Papers Chair

Patrick Eugster
Purdue University

Onward! Papers Chair

Gail Murphy
University of British Columbia

Onward! Essays Chair

Guy Steele
Oracle Labs

<http://splashcon.org>
info@splashcon.org



Computing Reviews

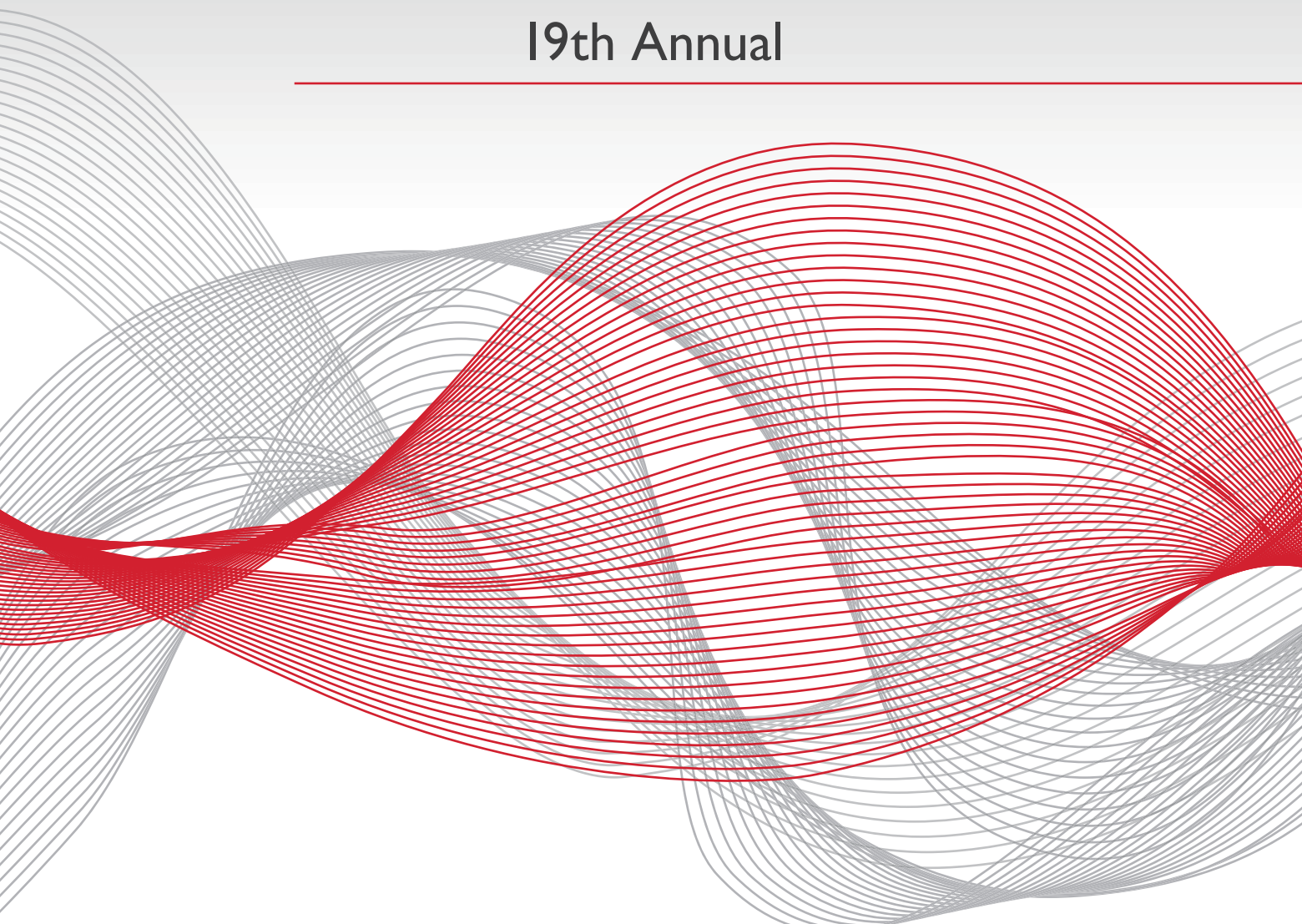


Association for
Computing Machinery

ThinkLoud

BEST OF COMPUTING

19th Annual



BEST REVIEWS
NOTABLE BOOKS & ARTICLES

Available Online & in Print

www.computingreviews.com/best19