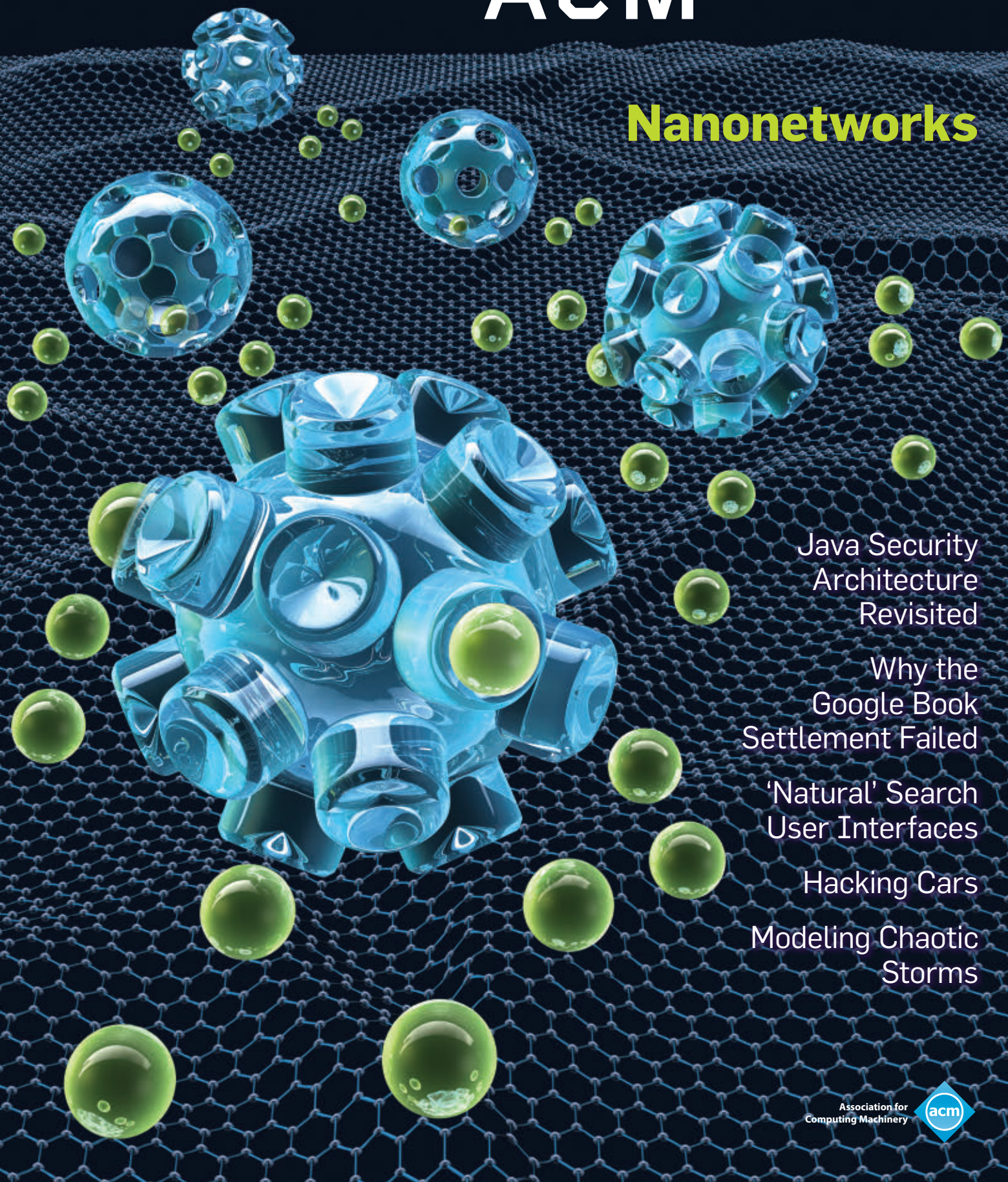
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

11/2011 VOL.54 NO.11



Nanonetworks

Java Security Architecture Revisited

Why the Google Book Settlement Failed

'Natural' Search User Interfaces

Hacking Cars

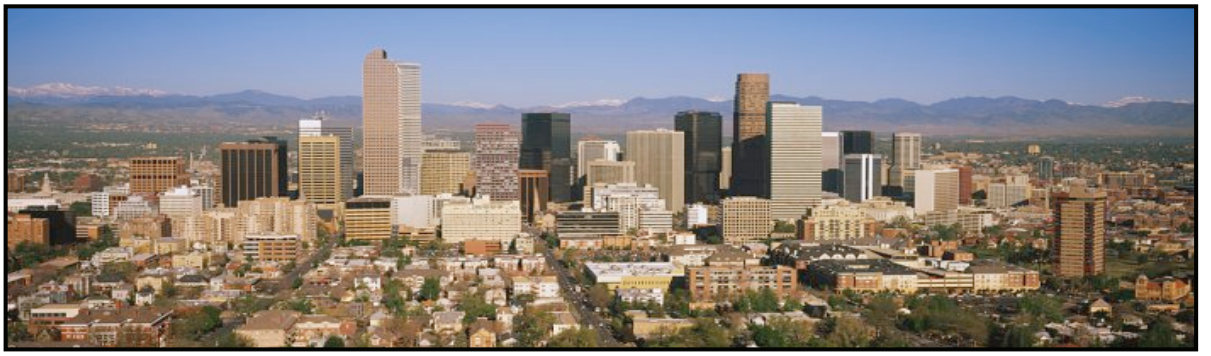
Modeling Chaotic Storms

Association for Computing Machinery

CALL FOR PARTICIPATION

CTS 2012

Denver, Colorado, USA



The 2012 International Conference on Collaboration Technologies and Systems

May 21 – 25, 2012
The Westin Westminster Hotel
Denver, Colorado, USA

Important Dates:

Paper Submission Deadline	**December 30, 2011**
Workshop/Special Session Proposal Deadline	**November 15, 2011**
Tutorial/Demo/Panel Proposal Deadline	**January 08, 2012**
Notification of Acceptance	**February 01, 2012**
Final Papers Due	**March 01, 2012**

For more information, visit the CTS 2012 web site at:
http://cisedu.us/rp/cts12 or http://cts2012.cisedu.info/

In cooperation with the ACM, IEEE, IFIP

Previous A.M. Turing Award Recipients

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

Additional information on the past recipients of the A.M. Turing Award is available on: http://awards.acm.org/homepage.cfm?awd=140

ACM A.M. TURING AWARD NOMINATIONS SOLICITED

Nominations are invited for the 2011 ACM A.M. Turing Award. The Association's oldest and most prestigious award is presented for contributions of a technical nature to the computing community. Although the long-term influences of the nominee's work are taken into consideration, there should be a particular outstanding and trendsetting technical achievement that constitutes the principal claim to the award. The award carries a prize of $250,000 and the recipient is expected to present an address that will be published in an ACM journal. Financial support of the Turing Award is provided by the Intel Corporation and Google Inc.

Nominations should include:

1) A curriculum vitae, listing publications, patents, honors, other awards, etc.

2) A letter from the principal nominator, which describes the work of the nominee, and draws particular attention to the contribution which is seen as meriting the award.

3) Supporting letters from at least three endorsers. The letters should not all be from colleagues or co-workers who are closely associated with the nominee, and preferably should come from individuals at more than one organization. Successful Turing Award nominations usually include substantive letters of support from a group of prominent individuals broadly representative of the candidate's field.

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

Nominations should be sent electronically by November 30, 2011 to: Jennifer Chayes c/o mcguinness@acm.org

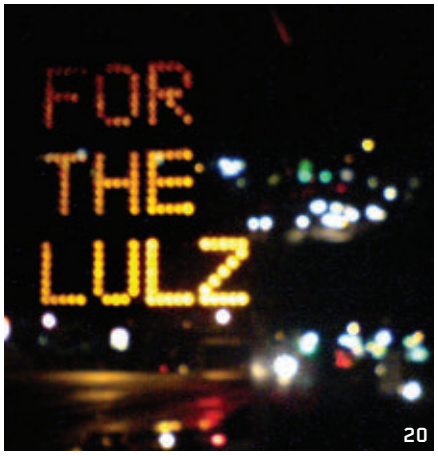
Association for Computing Machinery

COMMUNICATIONS OF THE ACM

Departments

Last Byte

News

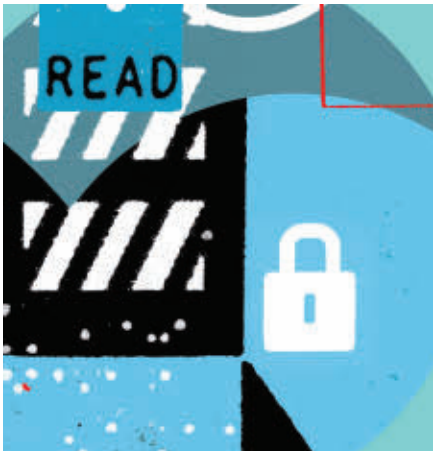


Viewpoints

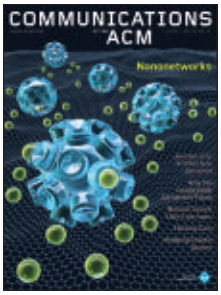
PHOTOGRAPH BY ANDREW ALBOSTA

11/2011
VOL. 54 NO. 11

Practice



Articles' development led by acmqueue
queue.acm.org



About the Cover: Nanotechnology enables the creation of devices the size of a human cell, prompting the need for new protocols. This month's cover story traces the state of nanonetworks as the new frontier in communications. Brooklyn-based artist Giacomo Marchesi (who also goes by the name James Gray) worked closely with the authors of the article to capture the essence of the nanomachines and nano-particles in 3D. For more on Marchesi's work, see http://www.giacomomarchesi.com.

ILLUSTRATOINS BY ALEX WILLIAMSON, GARY MARSH

Contributed Articles

Review Articles

Research Highlights

Association for Computing Machinery
Advancing Computing as a Science & Profession

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.

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.

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

Association for Computing Machinery

Printed in the U.S.A.

DOI:10.1145/2018396.2018397

Moshe Y. Vardi

Is Moore's Party Over?

The retirement of the U.S. space shuttle fleet brought a *Wall Street Journal* columnist to lament that "Mankind Nears the End of the Age of Speed." The article also mentioned the retirements of the supersonic Concorde and the SR-71 Blackbird spy plane, as well as Boeing's abandonment of its concept airliner, the Sonic Cruiser. "The human race is slowing down," complained the author.

Reading that article made me pause to reflect on the slowdown of computing. For almost 50 years we have been riding Moore's Law's exponential curve. Oh, what a ride it has been! No other technology has ever improved at a geometric rate for decades. It has been nothing short of a wild party. But exponential trends always slow down eventually, and the end of "Moore's Party" may be near.

I am not betting here against "Moore's Law." Such a bet is a well-known sucker bet. But Moore's Law is often "over-stood." One often reads how Moore's Law predicts the ongoing improvement in microprocessor speed or performance. But Moore's Law says nothing about speed or performance; Moore's 1965 paper was strictly about the exponential increase in transistor density on a chip. How can increased transistor density be translated to improved compute performance? After all, it has really been the improvement in performance that has changed the world around us dramatically since the beginning of the computer age. Indeed, over the past 50 years the computer industry faced a dual challenge. First, it had to keep marching to the drum of Moore's Law, which turned from an astute observation to a self-fulfilling prophecy. Second, it had to translate an increase in transistor density to an increase in compute performance.

This translation was accomplished in two ways. First, Moore's Law is underlain by the continued scaling down of transistor size, postulated by IBM researchers in 1974. This enabled transistors to be switched faster and faster, increasing microprocessor frequency. Second, and crucially important, has been the ability of computer architects to harness the power of transistor parallelism to speed up the execution of sequential programs by using bit-level and inter-instruction parallelism.

This unstoppable march hit a wall in May 2004, when Intel canceled its Tejas and Jayhawk microprocessor projects because of heat problems caused by high power consumption. Thus, just as the world economy is struggling with the energy crisis, the computer industry is struggling with its own energy crisis. Dealing with this crisis has been the major challenge for the industry for the last few years. A July 2008 *Communications*' article by Mark Oskin entitled "The Revolution Inside the Box" pointed out that the performance curve of microprocessors almost flattened in 2004, and concluded, "No longer is the road ahead clear for microprocessors." A May 2011 article "The Future of Microprocessors," by Shekhar Borkar and Andrew Chien, declared that "Energy efficiency is the new fundamental limiter of processor performance," and asserted that "Moore's Law continues but demands radical changes in architecture and software."

There are those, however, who argue that neither architecture nor software can be the solution. Provocatively titled "Dark Silicon and the End of Multicore Scaling," an ISCA'11 paper by H. Esmaeilzadeh et al. argues that energy is the fundamental barrier. Ultimately, improved performance requires more transistors to work faster in parallel, consuming more and more power. The paper predicts that as we continue to increase transistor density on a chip, an increased fraction of these transistors will have to be powered down and stay "dark." This means that even for highly parallel workloads we may see performance improvements lower than 20% per product generation.

While these predictions are a matter of ongoing debate, it is not too early, I believe, to start reflecting on their implications. For decades, the IT industry's business model has been predicated on double-digit annual performance improvements. I believe the next trend, which has already begun, is the commoditization of compute cycles. This will put inexorable pressure on profit margins of hardware vendors, bringing tremendous change to the computer industry, but it will make computing cheaper and more ubiquitous. The explosion of mobile devices, faced with their own energy challenges, is evidence of the force of this trend.

Peering further into the future, new physical phenomena, such as graphene and plasmonics, will replace today's dominant CMOS technology, unleashing a new age of compute-performance improvements. Remind me to write about this in 2020!

Moshe Y. Vardi, EDITOR-IN-CHIEF

DOI:10.1145/2018396.2018398

Justice for Jahromi

The committee of Concerned Scientists, the American Association for the Advancement of Science human rights and science program, and Scholars at Risk are deeply concerned over the fate of Masaud Jahromi, chairman of the Department of Computer Science and Engineering at Ahlia University in Bahrain. (Jahromi earned a Ph.D. in telecommunication networking from University of Kent at Canterbury in the U.K.)

Scholars at Risk, an independent non-profit organization affiliated with New York University, has learned Jahromi was arrested and taken from his home at 2:30 a.m., April 14, 2011, and first held in Al Galaa Prison, then transferred to the Dry Dock Prison, where he has been detained since the end of April.

Contacts in Bahrain familiar with Jahromi's situation, as received by Scholars at Risk, report the police broke into Jahromi's house, threatened and harassed members of his family, confiscated the family's laptops, and beat Jahromi before taking him away to an undisclosed location. He was then denied access to his family for more than a month. Reports also indicate Jahromi is not receiving medical treatment for serious, diagnosed conditions, including Hepatitis C.

The nature of Jahromi's arrest and subsequent detention without access to medical care and family suggests disregard for international standards of due process and fair trial and detention, as guaranteed in the Universal Declaration of Human Rights and the International Covenant on Civil and Political Rights, to which Bahrain has acceded. Taking into account reported arrests of scholars in Bahrain following the pro-democracy protests in February and March, Jahromi's detention further suggests a wider attempt to intimidate intellectuals and limit academic freedom in Bahrain.

The Committee of Concerned Scientists (I am its Vice-Chair, Computer Science) urges Bahrain to uphold its obligations under international law with regard to Jahromi and intervene to ensure his well-being—including regular access to family, legal counsel of his choosing, and medical treatment—pending his earliest release.

I urge my ACM colleagues to join our effort in advocating Jahromi's scientific freedom and human rights. Please send letters of support to:

His Highness Shaikh Khalifa Bin
Salman Al Khalifa
Prime Minister
Ministry of Foreign Affairs
P.O. Box 547,
Manama, Kingdom of Bahrain

For more on Jahromi and others to whom you may write, please see Scholars at Risk http://scholarsatrisk.nyu.edu/Education-Advocacy/Alerts-Scholars-in-Prison.php

Jack Minker, College Park, MD

For Copyright, Require No Deliberate Action

Though I always find Pamela Samuelson's "Legally Speaking" Viewpoints valuable and usually agree entirely, I found myself disagreeing strongly with "Too Many Copyrights?" (July 2011). Acquiring rights to one's own creative works should not require any kind of deliberate action. Creative works should be automatically one's own exclusive property unless and until one deliberately waives or assigns those rights. Consider four examples of injustice that could result:

Naive. Your child publishes something notable, perhaps a poem, short story, or painting, on a social network, blog, or school bulletin board, unaware of copyright, and makes no copyright claim. Someone else appropriates the work and makes money from it, perhaps by including it in an anthology. Your child gets nothing and has no rights to the work.

Lack of knowledge. A person in, say, rural Africa, writes and performs a world-class song or other piece of music. Due to the norms of the local culture, such a person lacks awareness of even the notion of copyright and its worldwide legal implications, making no copyright claim. Someone else appropriates the work and makes a significant amount of money from it. The original composer/performer earns nothing and has no rights. Historical examples of such appropriation by collectors and publishers include taking from composers and performers in rural North America from the 1920s to the 1960s.

Ordinary human error. A person in the developed world, aware of copyright, nevertheless by accident fails to attach a proper copyright notice to a book, paper, or other artistic work. Again, someone else appropriates the work for profit, leaving the author/creator with no rights or benefit.

Expectation of privacy. A person keeps a private diary or journal, and, intending to never let it see the light of day, does not include a copyright notice, then loses the work, after which it ends up in the hands of a publisher who then publishes it for profit. Not only does the author have no rights or benefit, the author may be greatly embarrassed when the content goes public, yet lacks recourse.

Suppose I make a statue and place it on my front lawn. Must I include a claim of physical ownership? If I don't, can you walk up and simply take it away? What if I don't include a claim of intellectual ownership? Can you simply walk up and scan it with a 3D laser scanner, then make and sell bronze copies? How are the rights of physical and intellectual ownership different?

Copyright to creative works should automatically reside with their creators, with no action required by them. The alternative creates asymmetry between those with power, money, or special knowledge and those without. Do we really intend to give the sophisticated or unscrupulous (whether individuals or corporations) rights to appropriate the creative works of others?

James Prescott, Calgary, Canada

Pamela Samuelson's Viewpoint "Too Many Copyrights?" (July 2011), along with practically all other coverage of the subject of copyright, seems to be staring at the problem without actually seeing it. That problem isn't the law but the concept that the protected rights of the creator and the invention itself are equivalent, linked commodities.

Copyright is protection granted to the originator (whether a team or an individual) of a novel idea. The confusion comes from two daisy-chain errors: first, commoditizing copyright, so it can be bought and sold; and, second, having commoditized a personal "right" so the legal system then treats it as a commodity, even though its advocates scream it penalizes the creator.

We can't have it both ways.

People object to paying copyright/patent fees that never (or trivially) trickle down to the actual originator and which is supposed to be the point of the law. Another problem is the add-on protections, as in 70-year "lives" for patents and copyrights. I say let protection die with the creator. Why let it be used to line the pockets of corporations or descendants who haven't created anything? Moreover, rewarding people for mere proximity to genius is common but unconscionable.

As for "corporate" copyrights or patents, since when did a company invent anything? If patents are not fungible assets, companies wouldn't buy them, and the actual rewards would go to the real originators whose names appear on the patent's bottom line.

The argument that companies need copyright protection to remain competitive is specious. Companies provide an environment that supports creativity, allowing creative people to create neat stuff. Their alternative would be to simply not provide such an environment at all. The straw-man argument is that if I don't do it, nobody will. The reality is that someone can always turn an advantage into a profit without the legal system paving the road for them with gold.

Turning "protection" of intellectual property into something that can be sold is absurd. The original (presumably novel) idea can certainly be sold and should be, but letting me sell you both my idea and my insulation from competition at the same time is the root of the problem. The idea is mine; how to make money with it is yours. If you can't, don't buy it.

Corporations and lawyers have no business making a profit off protection granted to another person or persons with clever ideas. The original intent of copyright or patent protection was the creators get to pick who can recreate their innovations and under what conditions. I see no evidence in the original intent of copyrights or patents that the associated legal protection should follow with the idea itself.

David Byrd, Arlington, VA

Author's Response:

Prescott assumes my proposal would hurt authors and artists who are ignorant of or fail to comply with formalities, such as registration of copyrights, but I was not suggesting that works should automatically end up in the public domain for noncompliance with formalities. Rather, works should be protected against commercial exploitations, even when their authors have not registered their works.

It's just that some rights and remedies should be available to those who have registered, and not to those who haven't. Inventors don't get automatic protection from patent law; they have to apply for protection, so why treat authors differently? Copyright regimes around the world have had formal requirements in the past, and some still do.

Moreover, virtually every other property regime has formalities, too, such as registration of one's car; people somehow adapt to them.

Byrd raises questions about whether corporations are inventive and need intellectual property protection to remain competitive. There are surely uninventive companies, but research teams at Apple, IBM, and Microsoft, among others, have come up with significant innovations that have improved our lives. These companies rely on copyright to protect software as a means to recoup investments in developing programs, although they also rely on other intellectual property, such as trademarks and trade secrets, as well as on first-mover advantages. If other companies could just copy their programs and sell them in competition with their developers, it would be tough to recoup such investments.

I say corporate copyrights are a net positive for society, as long as the scope of protection is not too extensive.

Pamela Samuelson, Berkeley, CA

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.

Distinguished Speakers Program

talks by and with technology leaders and innovators

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

Need a Great Technical Speaker for Your Next 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 Speaker Program (DSP) and deliver compelling and insightful content to your audience at a remarkably reasonable price. Our program features renowned thought leaders in academia, industry and government, speaking about the topics that matter most in the computing and IT world today. Our booking process is simple and convenient, please visit us at: **www.dsp.acm.org**.

The ACM Distinguished Speaker 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 & 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 & 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 & Universities

DSP speakers represent a broad range of companies, colleges and universities, including: IBM, Microsoft, BBN Technologies, Raytheon, Sony Pictures Imageworks, National Institute of Standards and Technology, Lawrence Livermore National Laboratory, Siemens Information Systems Bangalore, Stanford University, University of Pennsylvania, University of British Columbia, Georgia Tech, Carnegie Mellon, UCLA, McGill University, Tsinghua University and many more.

Topics for Every Interest

Over 250 lectures are available from nearly 100 different speakers with topics covering Software Engineering, High Performance Computing, Human Computer Interaction, Artificial Intelligence, Gaming, Mobile Computing, and dozens more. Some of our most popular lectures include:

- Electronic Voting in the 21st Century
- Software Engineering Best Practices
- Software Under Siege: Viruses and Worms
- Spatial Databases and Geographic Information Systems
- Careers in Computing – How to Prepare and What to Expect

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.

To select a speaker for your next event, get started today at www.dsp.acm.org.
If you have questions, please send them to acmdsp@acm.org.

Microsoft Research

The DSP is sponsored, in part, by Microsoft Europe

Association for Computing Machinery

Advancing Computing as a Science & Profession

DOI:10.1145/2018396.2018399

In the Virtual Extension

To ensure the timely publication of articles, Communications *created the Virtual Extension (VE) to expand the page limitations of the print edition by bringing readers the same high-quality articles in an online-only format. VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on merit. The following synopses are from articles now available in their entirety to ACM members via the Digital Library.*

viewpoints article

DOI: 10.1145/2018396.2018423

Information Seeking: Convergence of Search, Recommendations, and Advertising

Hector Garcia-Molina, Georgia Koutrika, and Aditya Parameswaran

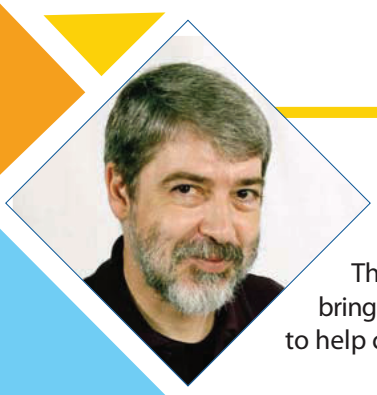
All of us are faced with a "deluge of data" in our workplaces and our homes: an ever-growing World Wide Web, digital books and magazines, photographs, blogs, tweets, email messages, databases, activity logs, sensor streams, online videos, movies and music, and so on. Thus, one of the fundamental problems in computer science has become even more critical today: how to identify objects satisfying a user's information need. The goal is to present to the user only information that is of interest and relevance, at the right place and time.

At least three types of information-providing mechanisms have been developed over the years to satisfy user information needs: A search mechanism takes as input a query that describes the current user interests; a recommendation mechanism typically does not use an explicit query but rather analyzes the user context and if available, a user profile; an advertisement mechanism is similar to a recommendation mechanism, except that the objects presented to the user are commercial advertisements, and financial considerations play a central role in ranking.

The authors argue that these mechanisms are not that different to begin with, and designing these three mechanisms making use of this commonality could lead to significant benefits. All three mechanisms share the same common goal: matching a context (which may or may not include an explicit query) to a collection of information objects (ads, product descriptions, Web pages, and so forth). The way context can be described varies, but there is no fundamental reason why one object type needs to be associated with one mechanism type. Similarly, the way matches are determined and ranked varies, but again, there is no fundamental reason why approaches cannot be used across mechanisms.

After reviewing some of the key concepts that historically evolved for search, recommendations, and advertisement, the authors explain why they believe a convergence of these technologies would be more useful today than in the past.

ACM, Advancing Computing as a Science and a Profession



Dear Colleague,

The power of computing technology continues to drive innovation to all corners of the globe, bringing with it opportunities for economic development and job growth. ACM is ideally positioned to help computing professionals worldwide stay competitive in this dynamic community.

ACM provides invaluable member benefits to help you advance your career and achieve success in your chosen specialty. Our international presence continues to expand and we have extended our online resources to serve needs that span all generations of computing practitioners, educators, researchers, and students.

ACM conferences, publications, educational efforts, recognition programs, digital resources, and diversity initiatives are defining the computing profession and empowering the computing professional.

This year we are launching Tech Packs, integrated learning packages on current technical topics created and reviewed by expert ACM members. The Tech Pack core is an annotated bibliography of resources from the renowned ACM Digital Library – articles from journals, magazines, conference proceedings, Special Interest Group newsletters, videos, etc. – and selections from our many online books and courses, as well an non-ACM resources where appropriate.

BY BECOMING AN ACM MEMBER YOU RECEIVE:

Timely access to relevant information

Communications of the ACM magazine • ACM Tech Packs • *TechNews* email digest • Technical Interest Alerts and ACM Bulletins • ACM journals and magazines at member rates • full access to the *acmqueue* website for practitioners • ACM SIG conference discounts • the optional ACM Digital Library

Resources that will enhance your career and follow you to new positions

Career & Job Center • The Learning Center • online books from Safari® featuring O'Reilly and Books24x7® • online courses in multiple languages • virtual labs • e-mentoring services • *CareerNews* email digest • access to ACM's 36 Special Interest Groups • an acm.org email forwarding address with spam filtering

ACM's worldwide network of more than 100,000 members ranges from students to seasoned professionals and includes many renowned leaders in the field. ACM members get access to this network and the advantages that come from their expertise to keep you at the forefront of the technology world.

Please take a moment to consider the value of an ACM membership for your career and your future in the dynamic computing profession.

Sincerely,

Alain Chesnais

President
Association for Computing Machinery

Association for Computing Machinery

Advancing Computing as a Science & Profession

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD10

You can join ACM in several easy ways:

Online	Phone	Fax
http://www.acm.org/join	*+1-800-342-6626 (US & Canada)* *+1-212-626-0500 (Global)*	*+1-212-944-1318*

Or, complete this application and return with payment via postal mail

Special rates for residents of developing countries:
http://www.acm.org/membership/L2-3/

Special rates for members of sister societies:
http://www.acm.org/membership/dues.html

Please print clearly

Name

Address

City | State/Province | Postal code/Zip

Country | E-mail address

Area code & Daytime phone | Fax | Member number, if applicable

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

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:
http://www.acm.org/serving/ethics.html

choose one membership option:

PROFESSIONAL MEMBERSHIP:

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

STUDENT MEMBERSHIP:

- o **ACM Student Membership: $19 USD**
- o **ACM Student Membership plus the ACM Digital Library: $42 USD**
- o **ACM Student Membership PLUS Print *CACM* Magazine: $42 USD**
- o **ACM Student Membership w/Digital Library PLUS Print *CACM* Magazine: $62 USD**

All new ACM members will receive an ACM membership card.
For more information, please visit us at www.acm.org

Professional membership dues include $40 toward a subscription to *Communications of the ACM*. Student membership dues include $15 toward a subscription to *XRDS*. Member dues, subscriptions, and optional contributions are tax-deductible under certain circumstances. Please consult with your tax advisor.

RETURN COMPLETED APPLICATION TO:

Association for Computing Machinery, Inc.
General Post Office
P.O. Box 30777
New York, NY 10087-0777

Questions? E-mail us at acmhelp@acm.org
Or call +1-800-342-6626 to speak to a live representative

Satisfaction Guaranteed!

payment:

Payment must accompany application. If paying by check or money order, make payable to ACM, Inc. in US dollars or foreign currency at current exchange rate.

o Visa/MasterCard o American Express o Check/money order

o Professional Member Dues ($99 or $198) $ _____

o ACM Digital Library ($99) $ _____

o Student Member Dues ($19, $42, or $62) $ _____

Total Amount Due $ _____

Card # Expiration date

Signature

BLOG@CACM

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.

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

DOI:10.1145/2018396.2018400 http://cacm.acm.org/blogs/blog-cacm

In Support of Open Reviews; Better Teaching Through Large-Scale Data Mining

Bertrand Meyer writes about his long-standing decision not to provide anonymous reviews. Greg Linden considers how educational practices could be improved through the data mining of students' schoolwork.

Bertrand Meyer
"Fixing the Process of Computer Science Refereeing"
http://cacm.acm.org/blogs/blog-cacm/100030
October 20, 2010

At ECSS 2010, the annual meeting of Informatics Europe, we heard a fascinating keynote by Moshe Vardi, editor-in-chief of *Communications of the ACM*, titled "The Tragedy of the Computing-Research Commons." Professor Vardi talked about the importance of engaging in activities that benefit the community even if they bring no huge immediate reward to the individuals who participate in them. He lamented the degradation of the computer science culture due, among other causes, to shoddy refereeing practices.

Lamenting about the reviewing process is common, and everyone has horror stories. Yet the simple solution is almost never considered: Turn the default refereeing mode to open, rather than anonymous.

BERTRAND MEYER
"Refereeing should be what it was before science publication turned into a business: scientists giving their polite but frank opinion on the work of other scientists."

Some cases may still justify anonymity, but they should be the exception, calling for a specific justification. Refereeing should be what it was before science publication turned into a business: scientists giving their polite but frank opinion on the work of other scientists. Anonymity just encourages power games, back stabbing, and, worst of all, poor quality: Since no one can call your bluff, you are not encouraged to be a good referee. Of course, many people do an excellent job anyway, but they do not necessarily prevail. In the highly competitive world of computer science publications—conference publication, in particular, with its schedule pressures—one incompetent but arrogant negative review typically outweighs five flattering and carefully considered analyses.

By revealing who you are, you force yourself to write reviews that you can defend.

More than two decades ago I started refusing to do anonymous reviews. This stance may not have only brought me new friends (which may not be a big deal as I am not sure people who hate you because you found flaws in one of their papers are worth having as scientific friends), but it has certainly made me a better reviewer. In fact, it did bring me "some" friends—people who are grateful for having gained new insights, positive or negative, into their own work.

A more complete discussion and rationale can be found in this Web page, http://se.ethz.ch/~meyer/publications/online/whysign, to which I regularly refer editors asking for reviews. That text, written several years ago, is verbose

and should be rewritten, but it does include the basic analysis.

The decision to perform open refereeing was personal and, until now, I have always refrained from proselytizing. Seeing the degradation in refereeing, however, I believe such reserve is no longer appropriate. Establishing open refereeing as the default strategy is the first step toward fixing the flawed culture of computer science refereeing.

Greg Linden
"Massive-Scale Data Mining for Education"
http://cacm.acm.org/blogs/blog-cacm/101489
November 10, 2010

Let's say, in the near future, tens of millions of students start learning math using online computer software. Our logs fill with a massive new data stream, millions of students doing billions of exercises, as the students work.

In these logs, we will see some students struggle with some problems, then overcome them. Others will struggle with those same problems and fail. There will be paths of learning in the data, some of which quickly reach mastery, others of which go off in the weeds.

At Amazon.com a decade ago, we studied the trails people made as they moved through our Web site. We looked at the probability that people would click on links to go from one page to another. We watched the trails people took through our site and where they went astray. As people shopped, we learned how to make shopping easier for others in the future.

GREG LINDEN

"Let's say we have massive new logs of what these students are doing and how well they are doing. What would a big Internet company do with this data?"

Similarly, Google and Microsoft learn from people using Web search. When people find what they want, Google notices. When other people do that same search later, Google has learned from earlier searchers, and makes it easier for the new searchers to get where they want to go.

Beyond a single search, the search giants watch what people look for over time as they do many searches—what they eventually find or whether they find nothing, where they navigate to after searching—and learn to push future searchers onto the more successful paths trod by those before them.

So, let's say we have millions of students learning math on computers. Let's say we have massive new logs of what these students are doing and how well they are doing. What would a big Internet company do with this data? What would be the Googley thing to do with these logs? What would massive-scale data mining look like for students?

We could learn that students who have difficulty solving one problem would have trouble with another. For example, perhaps students who have difficulty with the problem $(3x - 7 = 3)$ have difficulty with $(2x - 13 = 5)$.

We could then learn of clusters of problems that will be difficult for someone to solve if they have the same misunderstanding of an underlying concept. For example, perhaps many students who cannot solve $(3x - 7 = 3)$ and similar problems are confused about how to move the -7 to the other side of the equation.

Also, we could discover the problems in that cluster that are particularly likely to teach that concept well, to break students out of the misunderstanding and then be able to solve all the problems they previously found so difficult. For example, perhaps students who have difficulty with $(3x - 7 = 3)$ and similar problems are usually able to solve that problem when presented first with the easier problems $(x - 5 = 0)$ and $(2x - 3 = 1)$.

Then we could learn paths through clusters of problems that are particularly effective and rapid for students. Teachers might think one concept should always be taught before another, but what if the data shows us different? What if we reorder the problems and students learn faster?

We could even learn personalized and individualized paths for effective and rapid learning. Some students might start on a generic path, show early mastery, and jump ahead. Others might struggle with one type of problem or another. Each time a student struggles, we will try them on problems that might be a path for them to learn the underlying concepts and succeed. We will know these paths because so many others struggled before, some of which found success.

As we experiment, as millions of students try different exercises, we forget the paths that consistently led to continued struggles, remember the ones that lead to rapid mastery, and, as new students come in, we put them on the successful paths we have seen before.

It would be student modeling on a heretofore unseen scale. From tens of millions of students, we automatically learn tens of thousands of models, little trails of success for future students to follow. We experiment, try different students on different problems, discover which exercises cause similar difficulties, and which exercises help students break out of those difficulties. We learn paths in the data and models of the students. We learn to teach. C

Bertrand Meyer is a professor at ETH Zurich.
Greg Linden is the founder of Geeky Ventures.

DOI:10.1145/2018396.2018401 Scott E. Delman

ACM Offers a New Approach to Self-Archiving

In last month's issue (p. 5), Publications Board co-chairs Ronald Boisvert and Jack Davidson highlighted some of the recent changes to ACM's Copyright Policy and at the same time introduced a new service launched several weeks ago. This new service, aptly named the ACM Author-Izer, is a unique link-based self-archiving tool that enables ACM authors to generate and post links on either their home page or institutional repository (IR) that will lead any visitors clicking on these links to a free and definitive version of the author's article archived inside the ACM Digital Library. The main goals of the service are to offer the computing community the greatest possible access to the definitive versions of published works and to empower ACM's authors to showcase the definitive versions of their work on their home pages or IRs in a way that is consistent with ACM's existing subscription model.

In addition to being among the first publishers to offer such a service, ACM is effectively jumping into the Open Access debate with a bold move that crosses the line many other publishers have been hesitant to cross in recent years regarding self-archiving. While we feel very strongly that the existing paid-subscription model coupled with fair and affordable pricing is in the best interest of both the computing community and ACM alike, we also believe strongly that it is in the best interest of the community to provide easy access to only one version of an author's work, and that this should be the definitive version. Increasingly, access to published work starts with a Google or Google-like search and the user is often given a long list of links to a particular work. In many cases, these links lead to earlier versions of an author's work or even accepted but not yet published versions. Such versions may be hosted on different sites and it becomes increasingly difficult for users to distinguish between immature pre-published works, accepted but not yet published works, and the definitive works. In addition to the potential confusion caused, the user is oftentimes not using the most "functional" version of the article containing important metadata, reference links, and figures or images that have been added by the publisher during the final stages of the publication process, nor is the author given valuable feedback in the form of usage statistics for how often their articles are being downloaded and read.

By creating a free and persistent link for the author to post on their own home page or IR and enabling real-time usage statistics to flow through to those pages from the DL archive itself, ACM is encouraging authors to do away with the multiple versions that tend to build up over time on multiple sites and point to the one definitive article that exists. ACM authors continue to enjoy all of the existing rights and benefits of authorship with ACM, but there is now an added benefit that we believe will be embraced by the community.

To "Author-Ize" your own published articles in the ACM Digital Library, simply go to your own ACM Author Profile Page inside the DL, sign in, and follow the comprehensive instructions on how to generate these persistent links. Once you've done this, please let us know what you think at portal-feedback@hq.acm.org. Thank you.

ACM Member News

INCREASING DIVERSITY WITH MARK GUZDIAL

Mark Guzdial is focused on a very human aspect of computer science—the lack of diversity among those pursuing computing, either academically or vocationally.

Improving opportunities for women, African-Americans, Latinos, and other groups isn't simply the right thing to do, it's also critical for computer science, says Guzdial, a professor in the College of Computing at the Georgia Institute of Technology.

"We design computer programs for people who do the designing," he notes. "If we're going to create programs that are valuable for a broad section of the public, we have to draw upon all of our society to seek out our designers."

Guzdial is director of Georgia Computes!, a program that is boosting diversity in computer science education by working with state schools and reaching out to local Girl Scout, YWCA, and other community groups. Today, due in large part to Georgia Computes!, nearly 40% of the high schools in Georgia employ a teacher who has participated in the program. Those high schools produce 56% of the state's introductory computer science students and a majority of its women and minority computer science students.

Earlier this year, Guzdial and Barbara Ericson, director of computing outreach in the College of Computing, received ACM's Karl V. Karlstrom Outstanding Educator Award, for their contributions to broadening participation in computing.

Guzdial is also trying to develop a computer science medium for instructing high school teachers that puts more focus on electronic and online tools. "We need e-books that take a multimodal and multisensory approach," he says, "a new kind of medium to develop more high school computer science teachers."

—Dennis McCafferty

Science | DOI:10.1145/2018396.2018402

Kirk L. Kroeker

Modeling Chaotic Storms

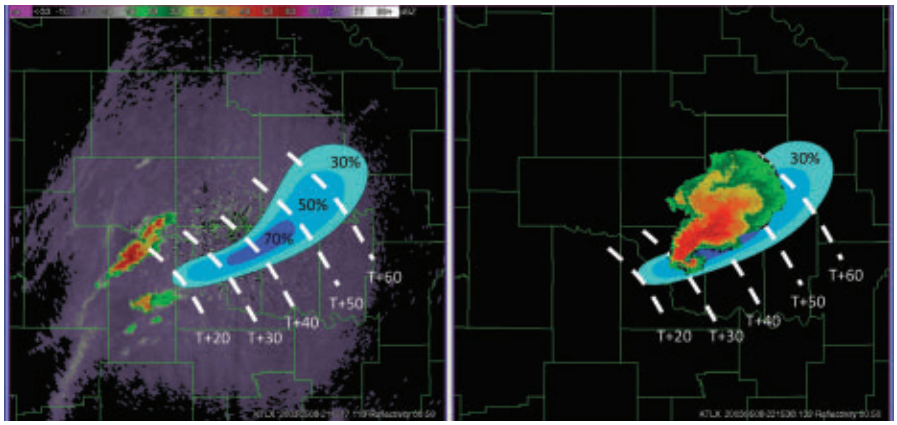
Scientists say improvements to extreme-weather prediction are possible with new weather models and a reinvention of the modeling technologies used to process them.

The warning time for the onset of extreme weather is far greater today than it was 20 years ago, when only a few minutes of warning could be given for tornadoes, and only half of them could even be predicted. Today, new data-collection technologies, such as Doppler radar and satellites, have improved the ability to identify and track hazardous weather. But scientists say further improvements to warnings' lead times will not come primarily from the physical systems that gather weather data, but from improving the modeling technologies used to process the data, and from improving the prediction models.

Even with the best data-collection technology, weather-tracking systems cannot be effective at predicting what might come next without refined prediction models that take into account advanced physics and other factors that make running such models a task that takes so long, even on the fastest supercomputers, that the results are not timely enough to be useful. Steve Koch, who was the director of the Global Systems Division at the U.S. National Oceanic and Atmospheric Administration's (NOAA's) Earth Systems Research Laboratory (ESRL) before becoming director of its National Severe Storms Lab (NSSL), is focused on developing advanced computing architectures to improve this situation.

"It looks like we are approaching an average lead time of about 14 minutes of warning for tornadoes, and we are reaching the limit of how far we can push detection with technology," says Koch. "We might get perhaps 20 minutes by using more advanced radar technologies, but ultimately if we want to get to a one-hour forecast of a tornado, we cannot do that right now just with radar."

Koch, who has been working at NOAA for more than a decade, says the organization is focusing extensively on global models, not only of weather but also of climate and the influence of oceans, land surfaces, ice and snow, aerosols, and other factors, including everything from anthropogenic effects on climate to invasive species in the Great Lakes. As a result, this focus has produced what Koch calls an "explosion of modeling" in the past few years. "There is great interest in integrating as many of these processes as possible into a coherent, scientific approach to the problem of forecasting the Earth system," he says.



In the Warn-on-Forecast system being developed at the National Severe Storms Laboratory, an emerging thunderstorm is observed by radar (left). That radar data is used as input for an ensemble of prediction models to determine the probability of a tornado during the next hour. The blue shading represents tornado probability, while the white dashed lines indicate the storm location in minutes from the present. Thirty minutes later (right), a thunderstorm is observed by radar over the area of greatest tornado probability, as predicted.

However, despite the proliferation of such models, the idea of a completely integrated approach, says Koch, is something that NOAA is still working out. "We don't have one gigantic model that solves all the problems," he says. Even so, one result of this intensive focus on modeling is that NOAA has made measurable progress at improving the ability to predict extreme weather more accurately. However, at NSSL, the focus of which is tornadoes, hail, and high winds, the difficulty of such predictions is acute.

Until recently, weather prediction has mostly relied on a few central prediction models, but that strategy has given way to running an ensemble of models, sometimes upward of 40 models, each with slightly different physics or other configurations. The idea is to create models that can capture the wide variability in the atmosphere and then combine them so there is sufficient spread in the characteristics of a complete ensemble. The ultimate goal of ensemble modeling, then, is to develop a broader representation of plausible outcomes while also producing useful estimates for forecasting.

One problem, however, is that even with an ensemble of models, the prediction results are limited by the quantity and quality of input data. In the U.S., for example, 142 weather radars cover the lower 48 states, with an average spacing of approximately 260 kilometers between them. Because of the Earth's curvature, more than 70% of the U.S. is not covered by radar in the lowest 1 kilometer of the atmosphere, which is, of course, where humans live. It is also where severe thunderstorms develop. "To overcome the Earth's curvature problem," says Koch, "you would need to develop a cost-effective, much-denser coverage system."

Until recently, weather prediction has largely relied on a few central prediction models, but that strategy has changed to running an ensemble of models, each with slightly different physics or other configurations.

Another challenge is the weather processes. What creates severe turbulence, for example, is far from being completely understood. "There are gaps in our understanding of processes within thunderstorms, and one of the biggest gaps is the physics of the precipitation process itself," says Koch. "We have an unsatisfactory understanding of what generates organized thunderstorms."

A Matter of Volume

Beyond these issues, which have drawn much attention by researchers in recent years, the sheer volume of data produced by an ensemble model configuration, which scientists now believe is the most effective way to predict extreme weather, is much more than the bandwidth of the most advanced networks can handle. In addition, while an ensemble of models can be tuned to the particular aspect of the weather it is designed to simulate, rendering the ensemble quickly enough to be useful for real-time forecasting remains a challenge, even with today's fastest supercomputers.

The power and cooling requirements are formidable for supercomputer-class systems, and the maintenance costs can be exorbitant. So Koch and other NOAA scientists are looking beyond traditional supercomputer systems and are exploring systems based on graphical processing units (GPUs). While GPUs are not general purpose, they do offer performance advantages over general-purpose CPUs for certain applications. NOAA has not made a decision yet about implementing GPU systems, but trials are in an advanced exploratory phase run at NOAA's ESRL.

Mark Govett, chief of the advanced computing lab at ESRL, has been di-

Technology

PC and Tablet Sales Forecast

Industry analysts are less optimistic now than earlier this year about PC sales for 2011. But the personal computer's woes have little to do with the public's enthusiasm for tablets as some believe, says Stephen Baker, vice president of industry analysis for The NPD Group.

Research company Gartner has trimmed its forecast for 2011 PC sales from 9.8% growth just three months ago to 3.8%, while rival research firm IDC lowered its forecast from 7.1% to 4.2%.

The NPD Group would not release specific numbers, but Baker characterizes U.S. PC sales as having been mostly flat to slightly down, mainly due to the weak economy. "And also because people bought record numbers of computers these last two years ever since the launch of Windows 7," Baker says. "Those new computers won't need to be replaced or upgraded for a while."

On the other hand, tablet sales—which have been dominated by Apple's iPads—have grown dramatically from a first-year small base, Baker says. There was no tablet market until the iPad was released in April 2010.

But, Baker adds, $600 iPads have had very little effect on the sales of less-pricey PCs that average under $500 for notebooks and about $500 for desktops. "Also, people typically buy tablets for different reasons than for PCs, mostly for content consumption, like watching movies, listening to music, and surfing the Net, while they buy PCs for content creation and productivity, such as editing pictures, creating movies, and managing finances."

Baker believes these trends will continue until at least 2012's second half when Microsoft is expected to unveil Windows 8. "That's when we foresee some changes in PC momentum," he says, "which will start to put some pressure on the iPad."

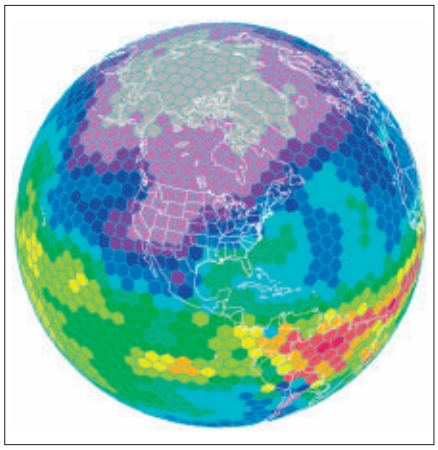
—Paul Hyman

recting these GPU trials. "We're always looking for cost-effective computing that can run research weather models," says Govett, who describes GPUs as the next generation of supercomputing technology. Govett says that, to run the new weather models, NOAA would need traditional CPU-based systems that would cost $75 to $150 million and would require building special data facilities to accommodate the power and cooling. "GPU systems with similar capabilities could be built for about $10 million, with no special building needed," he says.

As an example of the kind of cost savings involved with GPUs, Govett cites a test run with a next-generation model called the nonhydrostatic icosahedral model (NIM). The advanced computing lab at ESRL, which includes model developers, code-parallelization experts, and GPU researchers, calculates that more than 200,000 CPU cores would be needed to produce a NIM forecast close enough to real-time to be useful for prediction. The ESRL researchers, who began experimenting with GPUs in 2008, demonstrated that the NIM model could be run 25 times more quickly on GPUs than on traditional CPUs. (For an overview of new developments in high-end computing with GPUs, see "Supercomputing's Exaflop Target" in the August 2011 issue of *Communications*.)

While GPUs appear to be a promising alternative to traditional supercomputing, several challenges could prevent them from being adopted for weather modeling. For one, the code must be modified to run on GPUs. Govett and his team have developed their own compilers to convert their Fortran code into the language used on NVIDIA GPUs. As these compilers mature, Govett explains, the parallelization process will get easier. But, at least for now, Govett calls the work to parallelize models to run efficiently on GPUs a significant challenge. That code-parallelization difficulty is magnified with the new class of ensemble modeling that is expected to revolutionize weather prediction.

Single weather models, by themselves, present a significant challenge to high-performance systems capable of handling extreme workloads. Large ensembles consisting of many members (or models with different configurations) generate a range of forecasts that are then combined to produce a single more accurate forecast. At the finest scale needed for accurate weather prediction, these ensembles can be run only on the fastest supercomputers. "We recently ran a 20-member ensemble on the Oak Ridge Jaguar supercomputer, which was the largest supercomputer in the world until last year," says Govett. "That model required over 120,000 CPU cores, or basically half of the machine, and this was for one ensemble run."



A hexagonal grid used by several next-generation global weather models. This particular grid is based on a 480-kilometer model. The next generation of weather models, driven by GPU technology, will be run at a scale of 2 to 4 kilometers, making neighborhood-level accuracy possible.

Govett says NOAA does not have the processing power to run large ensemble models at a research level, let alone at an operational level where the models are run quickly enough for the predictions to be useful for forecasting. "Research computing, and to some extent climate forecasting, does not have the time constraint that operational weather forecasting does," says Govett. "In operations, weather models need to be run quickly or the information they produce will not be useful, particularly for severe weather where lives and property are at risk."

As for the future of ensemble modeling, Govett says that, by exploiting the parallelism available in GPU and multicore systems and rewriting existing code with new algorithms and solvers, ensemble models will be able to run at a global scale and generate weather and climate predictions with much better accuracy than can be achieved today. Despite the ongoing challenges facing Govett and other researchers working to refine weather modeling and find alternatives to traditional supercomputer-class systems so the more sophisticated ensemble models can move from research to operational use, Govett says he remains optimistic about the years ahead. "Model prediction continues to improve," he says. "This is particularly evident in the improved accuracy of hurricane track and intensity forecasts, severe weather predictions of flooding and tornadoes, and regional climate prediction."

Koch, for his part, says the future of weather prediction looks bright, indeed. But getting to the point where scientists are able to produce 60-minute warnings for extreme weather, he says, will be a major undertaking that will require enough computing power to run fine-scale ensemble models at an operational level. "That's my dream," he says. "It may be achievable within 15 years." C

Further Reading

Govett, M., Middlecoff, J., and Henderson, T.
Running the NIM next-generation weather model on GPUs, *Proceedings of the IEEE/ACM International Conference on Cluster, Cloud, and Grid Computing*, Melbourne, Victoria, Australia, May 17–20, 2010.

Henderson, T., Govett, M., Middlecoff, J., Madden, P., and Rosinski, J.
Experiences applying Fortran GPU compilers to numerical weather prediction models, *Proceedings of the 2010 Symposium on Application Accelerators in High Performance Computing*, Knoxville, TN, July 13–15, 2010.

Stensrud, D.J., et. al.
Convective-scale warn-on-forecast: A vision for 2020, *Bulletin of the American Meteorological Society 90*, 10, Oct. 2009.

Stensrud, D.J. and Gao, J.
Importance of horizontally inhomogeneous environmental initial conditions to ensemble storm-scale radar data assimilation and very short range forecasts, *Monthly Weather Review 138*, 4, April 2010.

Yussouf, N. and Stensrud, D.J.
Impact of high temporal frequency phased array radar data to storm-scale ensemble data assimilation using observation system simulation experiments, *Monthly Weather Review 138*, 2, Feb. 2010.

Based in Los Angeles, **Kirk L. Kroeker** is a freelance editor and writer specializing in science and technology.

Technology | DOI:10.1145/2018396.2018403 Alex Wright

Hacking Cars

Researchers have discovered important security flaws in modern automobile systems. Will car thieves learn to pick locks with their laptops?

Not so long ago, car thieves plied their trade with little more than a coat hanger and a screwdriver. New anti-theft technologies have made today's cars much harder to steal, but the growing tangle of computer equipment under the modern hood is creating new security risks that carmakers are just beginning to understand.

Ever since Toyota's well-publicized struggles with the computerized braking systems in its 2010 Prius hybrid cars, automotive computer systems have come under increasing scrutiny. In the last few years, researchers have identified a range of new, unexpected security flaws that could potentially affect large numbers of new cars. Given the specialized programming knowledge required to exploit these flaws, however, carmakers are still trying to gauge if these issues present a meaningful risk to ordinary drivers.

Last year, researchers Tadayoshi Kohno of the University of Washington and Stefan Savage of the University of California-San Diego announced the startling results of a two-year investigation into potential vulnerabilities in modern automotive computer systems.

The team initially explored whether they could compromise the onboard computer diagnostics port, a U.S. government-mandated feature in most modern cars. By inserting malicious code into the diagnostic software commonly found in auto repair shops and plugging a computer into the car's diagnostic port, they were able to stop the car's engine, lock the doors, and disable the brakes. More recently, they managed to remotely control a car by means of on-board Bluetooth or cellular services, thus demonstrating that a car could be controlled purely through wireless mechanisms.

"Our initial goal was to conduct a thorough, comprehensive analysis of a modern automobile," says Kohno. "This meant we wanted to study the brake controller, the engine controller, the light controller, the telematics unit, the media player, and so on. One of the biggest, most labor-intensive challenges was the sheer volume of components within the car."

Using an undisclosed hack, Kevin Finisterre was able to monitor a police car's video feed in real time.

Today's cars often contain myriad computer systems made by different manufacturers, making it difficult for any single component maker to identify every potential security exposure.

"To improve security one really desires a holistic view of all the components within the automobile," says Kohno, "but because of outsourced components it's hard for even the manufacturer to have that holistic view."

Despite the inherent difficulty of pinpointing security exposures in complex automotive systems, Kohno and Savage's work points to one conspicuous weak link: the onboard computer diagnostics port.

"Manufacturers could take steps to limit what someone might be able to do if they connect to the diagnostics port," says Kohno. He acknowledges, however, that the onboard port plays a crucial role in many cars. "One key challenge is to preserve the benefits but minimize the risks," he says.

Those risks seem destined to multiply as the number of network connections continues to grow, sometimes causing security exposures to crop up in unexpected places.

Take, for example, the humble tire. At the University of South Carolina, assistant professor Wenyuan Xu discovered that she could track the movement of cars by tapping into the RFID data stored in modern tire pressure monitoring systems from up to a distance of 40 meters.

Xu's team explored the proprietary communication protocols typically used to connect tire pressure sensors to onboard computers, and discovered that they could "listen" to the tire pressure sensors and use them to establish a connection with the onboard computers.

By capturing and decoding the tire sensor signals, the team was able to track the car's movements. They also established that they could send fake signals to trick the car computer into lighting up the low tire pressure warning light, regardless of the tire pressure. They were also able to inflict permanent damage to the tire pressure monitoring systems.

"An increasing number of wireless systems are installed in modern cars," notes Xu. "Wireless networks are known to be vulnerable to eavesdropping and packet injection."

Communication Breakdown

Xu's work points to a central problem with many modern automobiles: The Controller Area Network (CAN), which was originally designed to enable mi-

PHOTOGRAPH BY SERGEI CHUMAKOV / SHUTTERSTOCK.COM

crocontrollers to communicate with each other. As additional devices become connected to the CAN, the security exposures are multiplied. "It is a bad idea to trust any commands flowing on the CAN bus," says Xu. "More and more ECUs [Electronic Control Units] with wireless capabilities are added onto the CAN bus, which opens a door for remote attacks."

In a similar vein, researcher Kevin Finisterre of security consultancy Digital Munition recently drew headlines when he managed to compromise a police cruiser by taking advantage of security holes in the onboard networking system.

"I was working to help a police department vet its technology choices," recalls Finisterre, who has declined to identify the municipality that hired him. "The staff had several concerns with regard to the resiliency of their network to withstand an attack from a hacker."

Finisterre soon proved his client's hunch correct. After scanning several IP addresses known to be used by the city, he traced one of them back to a Linux machine installed inside one of the city's police cruisers. Using simple Telnet and FTP connections, he was able to access a streaming live video feed from the cruiser's onboard camera, as well as stored footage on a digital video recording device, and could upload, download, and delete footage stored on the car's onboard computer. At one point, Finisterre found himself monitoring the cruiser's video and audio feeds in real time as an officer responded to an incident.

Demonstrations like this may highlight potentially troubling flaws in modern automotive security, but how likely is it that ordinary car thieves will master these advanced computer science techniques in sufficient numbers to present a real threat to the average driver?

"To me, expertise is never a factor in determining threat level," says Finisterre. "Expertise can be gained either through rapid prototyping in a test environment or via simply social engineering someone who already has said experience. I think at this point in the game more targeted attacks may be occurring and being kept under wraps."

Finisterre acknowledges, however, that most of the threats remain largely

U.S. researchers have remotely controlled a car by means of on-board Bluetooth or cellular services, thus demonstrating that motor vehicles could be controlled purely through wireless mechanisms.

hypothetical. "The general population is most likely not currently exposed to much risk. If you, on the other hand, were in a position in which sensitive conversations are had in your vehicle I may be more concerned about the built-in system and its various data ingress points."

Xu agrees that the real threat to drivers is probably limited. Replicating her team's tire-hacking exercise would be expensive for most car thieves; each of her vehicle-tracking tools costs $1,500 to make. "It requires higher commitment, and thus imposes less risk," she explains. "Of course, the unit price can be further reduced, but it's still not a small number for regular consumers."

However, Xu isn't taking any chances. Her team won't release its tools to the public. "It is not impossible that some people driven by profit incentive could make and sell commodity products on eBay to unlock others' cars," she says.

While the practical risks may seem limited, nonetheless the automotive industry bears the ultimate responsibility—and potential legal liability—for ensuring the safety and security of its vehicles.

To date, computer security has lagged far down the list of automakers' business priorities. That may be starting to change, however, thanks to Toyota's 2010 Prius problems and a growing awareness of automotive security issues in the computer science community.

"Traditional computer security stra-tegies can drastically improve the computer security of modern automobiles," says Kohno. "I think at least some major industry players are now very aware of potential computer security concerns, and they are working very hard to try to mitigate those concerns."

To that end, a group of scientists from academia and industrial labs recently formed the Embedded Vehicle Safety Committee in an effort to set standards for computer security for future automobiles.

Not all researchers agree that the automotive industry is doing enough to address security issues, however. "There is some work going on," says Xu, "but not enough."

Finisterre agrees. "Bells and whistles often take precedence over security concerns," he says. In this respect, the automotive industry is no different from many other consumer-oriented industries, where marketing considerations and cosmetic features frequently take first priority. Until automakers see a meaningful impact on their bottom lines, computer security may continue to take a proverbial backseat. C

Further Reading

Checkoway, S., et al.
Comprehensive Experimental Analyses of Automotive Attack Surfaces, National Academy of Sciences Committee on Electronic Vehicle Controls and Unintended Acceleration, Washington, D.C., March 3–4, 2011.

Francillon, A., Danev, B., and Capkun, S.
Relay attacks on passive keyless entry and start systems in modern cars, *Proceedings of the 19th USENIX Security Symposium*, Washington, D.C., August 11–13, 2010.

Koscher, K., et al.
Experimental security analysis of a modern automobile, IEEE Symposium on Security and Privacy, Oakland, CA, May 16–19, 2010.

Rouf, I., et al.
Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study, *Proceedings of the 19th USENIX Security Symposium*, Washington, D.C., August 11–13, 2010.

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

Society | DOI:10.1145/2018396.2018404 Leah Hoffmann

Risky Business

Governments, companies, and individuals have suffered an unusual number of highly publicized data breaches this year. Is there a solution?

News about data breaches has been everywhere this year, with each new incident seemingly worse than the previous one. An attack on direct marketer Epsilon put an estimated 60 million records at risk. An intrusion into International Monetary Fund servers may have exposed confidential data about national economies. Attacks from Jinan, China sought to compromise the Gmail accounts of senior U.S. government officials and others, while Lockheed Martin suffered "a significant and tenacious attack" of an as-yet unspecified scope. And then there's Sony. One million passwords were stolen from Sony Pictures, 77 million accounts were compromised at the company's PlayStation network, and 25 million records were breached at Sony Online Entertainment. Taken together, it is the largest data breach on record. Although the annual number of breaches has fallen over the past few years, according to the Open Security Foundation's DataLossDB, the stakes are higher than ever—and there is a lot of work to be done to protect our sensitive data.

Data breaches are most often motivated by financial or political gain, but some hackers, like Lulz Security, attack companies for their own enjoyment or "for the lulz."

Attacks vary widely in scope and motivation. Some hackers work for their own disruptive pleasure, or "for the lulz," as one prominent group would have it. (Lulz Security, or LulzSec, has claimed responsibility for a number of prominent attacks this year and regaled the world with statements like "You find it funny to watch havoc unfold, and we find it funny to cause it.") Others are politically driven. However, most attackers are in it for the money.

"Revenge is a powerful motivator, but you need a return on your investment," explains Scott Ksander, chief information security officer at Purdue University, whose networks are tested with, by his estimate, an average of 300–500 incidents each week. "Some people develop and sell tool kits for exploits, and the current estimate holds that this industry is worth $100 million a year. Others mine and sell personal information. The more I know about you, the more effectively I can masquerade as you. It's the same concept that marketers use."

Attack methods are changing, too. A June report by Cisco Systems notes a steep decline in the number of mass attacks, such as self-propagating worms, DDoS attacks, and spam. Instead, attackers are turning to small, highly focused campaigns that are customized to specific user groups or even specific users. Spear phishers, for example, use data about the places people bank or shop to more effectively trick them into clicking on malware-infested email attachments or logging onto fraudulent Web sites.

The cost of mounting such attacks is not trivial—perpetrators must acquire quality victim lists, conduct background research, and generate sophisticated-looking email messages and Web sites—but conversion rates are high. Also, the payoff per victim—an average of $80,000, by Cisco's estimate—is up to 40 times greater than it is for mass attacks. Spear phishing is what victims of the Epsilon breach were warned against; ironically, it is also what caused the breach.

Meanwhile, the cost to affected companies, governments, and individuals is higher than ever. According to industry experts, Sony may spend up to $1 billion investigating the data breaches and dealing with the attendant problems. The Ponemon Institute, a Michigan-based research center, reports that each compromised record costs companies an average of $214 in 2010, up from $202 in 2009 and

PHOTOGRAPH BY ANDREW ALBOSTA

$197 in 2008. The cost to consumers varies by incident, but studies indicate that victims spend between $600 and $1,400 to resolve cases of identity theft, in addition to whatever money was stolen or scammed from them.

How, then, to respond? Basic principles are easy to agree on: Reduce software vulnerabilities, help users understand the risks, and minimize potential damage with secure network architectures and exploit mitigation techniques. Opinions differ, however, about how to accomplish those goals. Take software vulnerabilities, for example. Could new laws give companies an incentive to reduce them? Do we need new regulatory authorities? To what extent can we even expect to make bug-free software?

"Secure code is the first link in the chain," says Charlie Miller, chief security researcher at Accuvant Labs. "People say, 'We're human, we can't write perfect software.' But we're at 50% right now. We're not even close." Miller suggests that vendors create a cooperative fund to pay for the detection of bugs. "If a 17-year-old kid in Romania finds a bug, what will he do? Give it to the vendor for nothing or sell it to a black hat hacker?" Of course, some vendors pay for bugs, and TippingPoint's Zero Day Initiative buys others, but the overall economic incentives tend to favor would-be attackers.

Software Damages?

Legislation offers a different approach. In May, the European Commission introduced a proposal to hold companies liable for damages caused by faulty software. "We need to build trust so that people can shop around with peace of mind," European Union consumer commissioner Meglena Kuneva explained in a press release.

Historically, however, software-related damages have been difficult to prove. "There have been loads of lawsuits around data breaches. The majority has been class-action suits, and I'm not aware of any that succeeded," says Fred Cate, a law professor at Indiana University and director of the Center for Applied Cybersecurity Research.

Judges may dismiss suits because the victims are too diverse to be certified as a class or because there is no evidence they have been harmed as a direct result of the breach. (After all, Social Security and credit card numbers are stored in many places.) In 2006, for example, data aggregator ChoicePoint settled a U.S. Federal Trade Commission suit that was brought after it reported the theft of 163,000 user accounts. ChoicePoint established a $5 million restitution fund, but transferred most of it to the U.S. Treasury in 2008 after determining that only 131 consumers had presented valid claims.

According to industry experts, Sony may spend up to $1 billion investigating its multiple data breaches and dealing with the attendant problems.

Determining vendor responsibility can also be difficult. Say you have a bot on your computer that came through a plug-in to your Internet browser and compromised your operating system. Which company is at fault?

For the most part, very little has been done to legislate cybersecurity. Beyond the fields of health care and finance, the majority of proposals have stalled, and while U.S. and European lawmakers are now trying to standardize the patchwork of local security notification laws that require companies to alert the victims of a breach, the effectiveness of such rules may be limited if notifications become too common and consumers simply ignore them.

"Our legal response to this problem has been unimaginative," says Cate. "There is, for example, no incentive for cable companies to encrypt or secure the modems they place in customers' houses. As consumers, we have no incentive to use secure passwords, and companies have no incentive to make us use secure passwords. What if mo-

Internet

Daily Deals Shakeout

Internet sites that offer daily deals to consumers are dwindling as the industry, led by Groupon and LivingSocial, begins to shake out, according to a recent article in *The Wall Street Journal*.

The market has attracted many imitators of the leaders, but daily-deal-site aggregator Yipit.com reports that 170 of the 530 daily deal sites in the U.S. have either shut down or have been sold this year. Moreover, even deep-pocketed companies like Facebook and Yelp, which could capitalize on their large audiences, have scaled back on the service.

Why are daily deal sites disappearing? One of the biggest reasons is the shifting economics of the business, the *Journal* article reports. Although starting a daily deals business does not require much beyond creating a Web site and finding local merchants willing to offer a discount, and the expense of running a daily deals business have risen significantly as the industry has matured.

Specifically, it has become much more costly during the last two years to acquire subscribers who redeem daily deals, the article notes, citing executives at daily deal sites. While it did not take a large amount of marketing to win over early adopters, it now requires more spending to reach a broader audience and to stand out from the competition. Also, the sites now need to hire more salespeople to procure coupon offers from local merchants.

The *Journal* cites Groupon as an example of how costs have increased. Groupon spent about $8 to acquire each subscriber who redeemed a daily deal in the first quarter of 2010, according to regulatory filings. By the second quarter of 2011, that figure had nearly tripled to about $23.

Overall, Groupon spent $379 million in marketing in the first half of 2011, up from $35.5 million in the same period of 2010. Many smaller daily deal sites, however, simply do not have the resources to compete on that level.

—Bob Violino

"Secure code is the first link in the chain," says Charlie Miller, chief security researcher at Accuvant Labs. "People say, 'We're human, we can't write perfect software.' But we're at 50% right now. We're not even close."

bile phone companies got a dollar for every customer whose password they set? What if we had cybersecurity education programs, like we do for fire safety and AIDS?"

Information sharing is another idea that is proven easier to suggest than to implement. Because big breaches often evolve from smaller attacks, increased transparency about tactics and vulnerabilities could help contain the damage. With few exceptions, however, companies are slow to publicly acknowledge incidents, and they typically release little information about the attacks. Could they learn from academia, where security officers take a more collaborative approach?

"The Big Ten security people get together four times a year, and we email constantly," says Purdue's Ksander. "Generally, the communication is about sharing perspective on upcoming risks or solutions that an institution may be trying. We like to share both successes and failures. Learning about an attack at Iowa once helped clean up an event much faster here at Purdue."

A recent White House proposal called for the Department of Homeland Security to work more closely with businesses and manage information and incidents. Critics questioned the model, which would require companies to report all "significant" incidents and place the government at the hub of a hub-and-spoke information sharing model, but many other methods could also work.

"You could go by industry, especially where there are already regulators," suggests Cate. "You could do across-the-board reporting in annual reports and have the SEC [Securities and Exchange Commission] oversee it. Or you could do it in a less public and more detailed fashion, maybe on a state-by-state basis."

Of course, protection can never be perfect, and one thing nearly everyone agrees on is the need for organizations to work proactively to mitigate damage. "Good sysadmin practices have been the same for 10 years," says Art Manion, a vulnerability analyst at Carnegie Mellon University's Computer Emergency Response Team. "Things like firewalls that block all traffic that doesn't have a legitimate reason to cross a firewall or router or host. File system permissions on file server shares are another example—running services and programs, especially highly exposed stuff, with minimal privileges. Choosing where and how to store sensitive data, and how to transfer it, can also limit damage. If an attacker is able to steal encrypted data but not the decryption keys, damage is mitigated. These choices will largely depend on the needs and resources of individual sites, but I'm of the opinion that spending time, effort, and money on the basics is a better investment than security bells and whistles." C

Further Reading

Cate, F.H., Abrams, M.E., Bruening, P.J., and Swindl, O.
Dos and Don'ts of Data Breach and Information Security. The Centre for Information Policy Leadership, Richmond, VA, 2009.

Cate, F.H.
***Information Security Breaches: Looking Back & Thinking Ahead.* The Centre for Information Policy Leadership, Richmond, VA, 2008.**

Cisco Systems
Email Attacks: This Time It's Personal. Cisco Systems, San Jose, CA, 2011.

Center for Strategic and International Studies Commission on Cybersecurity for the 44th Presidency
***Securing Cyberspace for the 44th Presidency.* Center for Strategic and International Studies, Washington, D.C., 2008.**

DataLossDB
http://datalossdb.org/

Leah Hoffmann is a technology writer based in Brooklyn, NY.

Milestones

Computer Science Awards

The Association for the Advancement of Artificial Intelligence (AAAI), IEEE, and the Electronic Design Automation Consortium recently honored a select set of computer scientists.

AAAI SENIOR MEMBERS

AAAI awarded Senior Member status to nine distinguished AAAI members at the 25th AAAI Conference on Artificial Intelligence. The new Senior Members are Marie desJardins, University of Maryland Baltimore County; Hans W. Guesgen, Massey University; Tad H. Hogg, Institute for Molecular Manufacturing; Diane J. Litman, University of Pittsburgh; João Pavão Martins, Instituto Superior Tecnico, Technical University of Lisbon; Leora Morgenstern, SAIC; Ted E. Senator, SAIC; Ramasamy Uthurusamy, General Motors; and Holly Yanco, University of Massachusetts Lowell.

B. RAMAKRISHNA RAU AWARD

Yale N. Patt, an electrical and computer engineering professor at the University of Texas at Austin, is the recipient of the inaugural IEEE Computer Society B. Ramakrishna Rau Award, which recognizes significant achievements in the field of microarchitecture and compiler code generation.

PHIL KAUFMAN AWARD

C.L. David Liu, the William Mong honorary chair professor of computer science and former president of the National Tsing Hua University, received the Phil Kaufman Award for Distinguished Contributions to Electronic Design Automation (EDA) for his fundamental and seminal work in EDA.

DOI:10.1145/2018396.2018405

Hannes Tschofenig

Privacy and Security
Security Risks in Next-Generation Emergency Services

Sounding the alert on emergency calling system deficiencies.

In mid-June 2011, the National Emergency Number Association (NENA) approved the end-state architectural vision for Next Generation 9-1-1.[6] The "i3" architecture, as technical experts call it, presents a detailed architecture for key elements of the next-generation 9-1-1 systems, describing how networks and devices will eventually work together to enable voice, text, image, and data exchange between citizens and first responders.

It took NENA members years to complete the work on the i3 architecture and related specifications. Although NENA is the leading emergency services organization in North America, it did not develop the specifications alone. The i3 architecture borrows heavily from the standards developed by the Internet Engineering Task Force (IETF) on SIP, location, and emergency calling. It is not only the reuse of specifications that is significant but the mind-set acquired by NENA from the IETF: IETF's emergency services architecture follows the Internet application deployment model where applications may be provided by companies that are different than those providing Internet access. This assumption may not necessarily be surprising for Internet and smartphone application designers, but it is a signif-

An emergency communications technician responds to calls at the Emergency Communications Center in Arlington, VA.

PHOTOGRAPHH BY MATT MCCLAIN / THE WASHINGTON POST VIA GETTY IMAGES

icant change for anyone coming from the circuit-switched telephony world, which formed the current emergency services system for placing 9-1-1 and 1-1-2 emergency calls.

The regulatory community has also noticed the steady shift to Internet Protocols for all forms of communication. In early 2011, the Federal Communication Commission (FCC) issued a Notice of Inquiry (NOI) on the Framework for Next-Generation 911 Deployment[5] to solicit feedback on all forms of multimedia emergency calling.[a]

The characteristics associated with a deployed architectural model impact security. The fundamental security problem of emergency services is that these are services associated with a high cost, such as dispatching first responders like ambulances, law enforcement, fire department services, and the service must be available to all users, not just highly vetted ones. Consequently, there is much potential for misuse of the system, which unfortunately does occur. Yet the fact that the services must be universally available means there is little way to prevent such misuse.

Since the emergency services solutions are built on top of the existing communication architectures and protocols, they inherit the associated characteristics, including the security problems of Voice over IP, instant messaging (IM), and other forms of communication technologies. Yet despite these problems, from the point of view of economics it is unlikely to assume a separate end-to-end communication infrastructure will ever be deployed solely for use by emergency services.

False Emergency Calls

Among all the security challenges today's systems suffer most from so-called "false emergency calls," a form of denial-of-service attack. As the European Emergency Number Association (EENA), the European counterpart of NENA, has noted, "False emergency calls divert emergency services away from people who may be in life-threatening situations and who need urgent help. This can mean the difference between life and death for someone in trouble." EENA has attempted to define terminology and describe best current practices for dealing with false emergency calls,[3] which in certain European countries can be as high as 70% of all emergency calls. Reducing the number of bogus calls often represents a significant challenge, since emergency services authorities in most countries are required to answer every call (whenever possible). If there is no ability to associate the caller with a real-world person in case of misuse, then the ability to prosecute is limited. Due to requirements for supporting the so-called SIM-less emergency calls in many countries (emergency calls that are placed without a SIM card); calls from phones with pre-paid cards, or from public telephones make accountability difficult.

The fact that the services must be universally available means there is little way to prevent misuse.

While hoax call attacks typically lead to various negative results, they typically do not cause life-threatening situations. But a small percentage of these calls pose a significant risk. Most significantly, "swatting"—faking an emergency that draws a response from law enforcement (usually a SWAT team)—has the potential for causing life-threatening problems.[4]

The attack is fairly simple: the location system of today's telephony system performs a lookup using the telephone number as used by the caller. The obtained location information is then provided to the emergency number authorities for dispatch of first responders, in this case a SWAT team. Unfortunately, the caller's phone number can be modified.

A very similar attack can be used in IP-based emergency services systems.[13] In its simplest form, the adversary crafts location information and attaches it to an outgoing emergency call.

While there are various countermeasures, none are easy to deploy. When location information is obtained from the Internet access provider (as it is common for both fixed as well as cellular telecommunication emergency services systems), various identifiers must be linked to each other in order to obtain the physical location of the emergency caller. The proposed intermediate VoIP emergency services architecture developed by a U.K. standardization organization illustrates this mapping process in the example of a DSL network in Appendix E of an EENA operations document.[7] A weak link in the mapping process can be exploited. Similarly, when location measurements must be obtained, as those are often provided with the support of the end devices themselves (for example, from a GPS module). Naturally, an adversary in control of the end device is able to return fake measurement results and can thereby impact the obtained location.

An approach that focuses on the prosecution of those who misuse the service is difficult to accomplish in an IP-based emergency services solution as well. The challenges are primarily on the technical side but are a side effect of the deployment reality: strong identity proofing is not widely deployed by many VoIP/IM services nor is it deployed in the Internet in general. In-person identity proofing is expensive and by itself is not sufficient to provide a high level of assurances throughout the entire service life cycle (for example, as described in NIST SP 800-63).[8]

The Internet is global, and many application service providers operate their services everywhere. So, despite a perfect mapping between the digital identifier and a real-world person the solution to the problem will be dependent on the regulatory environment and the ability of law enforcement agencies in different countries to cooperate. For example, how easy will it be to hold a person located in country X using alice@example-service.com responsible for making a hoax call to an emergency service in country Y? This problem was identi-

a In their NOI response Barnes et al.[1] provided a high-level description of the IETF emergency services architecture and illustrated the main characteristics. A more technically minded reader may want to consult the original IETF specifications (see Rosen et al.[11] and Rosen and Polk[10]).

fied long ago and has surfaced again in the ongoing debate about the accountable Internet (for example, see Clark and Landau[2]).

The list of security threats in VoIP and IM systems is naturally quite long. Of course, research and standardization has also been ongoing, and various countermeasures have been developed and are waiting to be deployed. Some specific protocols also had to be developed to support emergency services, such as the Location-to-Service Translation (LoST) protocol that is used for routing emergency calls to the appropriate Public Safety Answering Points (PSAPs). These components introduce additional attack vectors.[12]

Conclusion

The work on the next-generation emergency services infrastructure is progressing with NENA and EENA leading the work in North America and Europe, respectively. With the baseline technical standards coming from the IETF, security threats have been investigated and documented, and technical countermeasures have also been developed. While many of these problems are being observed in today's emergency services system, it is likely the transition to an all-IP-based emergency services infrastructure will invite far more attacks. There are various barriers for dealing with these problems, namely:

- As long as attacks are still low (and the amount of IP-based emergency services is still low due to either the lack of regulation or unclear regulatory situation worldwide) there is no incentive to resolve the problem;
- Many Internet players seem to lack economic incentives to deploy infrastructure for a high level of assurances—an area that aims to be addressed by NSTIC but with uncertain success outcome at this point in time; and
- A lack of harmonization at the legal level. The Internet architectural model is not well enough understood by the regulatory community. The temptation to follow well-established patterns and to talk to their existing clientele is often too big; unfortunately that work style is not a match for today's ecosystem.

The last item is a particular area where further work is needed to bridge the gap between the policy and the technical community. Organizations developing technical standards are typically not ideally positioned to convey messages to policymakers on how responsibilities for the deployment have to be shared among the different stakeholders in the ecosystem and what nontechnical considerations need to be addressed. Due to their broader mandate, which includes training, certification, lobbying, and operational guidance, emergency services communities such as NENA and EENA are in an ideal position to close this gap.

In a nutshell, research and standardization have gone a long way toward providing the necessary building blocks. Now it is time for deployments to catch up. C

References

1. Barnes, R., Cooper, A., and Tschofenig, H. Technical considerations for next-generation 911—Comments in the matter of framework for next generation 911 deployment, PS Docket No. 10-255; http://fjallfoss.fcc.gov/ecfs/document/view?id=7021034355.
2. Clark, D., and Landau, S. Untangling attribution. In *Proceedings of a Workshop on Deterring CyberAttacks: Informing Strategies and Developing Options for U.S. Policy*, 2010.
3. EENA. False emergency calls EENA operations document; http://www.eena.org/ressource/static/files/2011_03_15_3.1.2.fc_v1.0.pdf.
4. FBI. Don't make the call—The new phenomenon of 'swatting,' Feb. 2008; http://www.fbi.gov/news/stories/2008/february/swatting020408.
5. Federal Communications Commission. Framework for next generation 911 deployment, PS Docket No. 10-255, December 21, 2010; http://www.fcc.gov/Daily_Releases/Daily_Business/2010/db1221/FCC-10-200A1.pdf.
6. NENA. NENA executive board approves end-state vision for next generation 9-1-1; http://www.nena.org/stories/technical/executive-board-approves-i3-standard.
7. NICC. VOIP—Location for Emergency Calls (Architecture), NICC ND 1638 Issue 1.1.2, March 2010; http://www.niccstandards.org.uk/files/current/ND1638%20V1.1.2.pdf.
8. NIST. DRAFT Electronic Authentication Guideline; Special Publication 800-63, June 2011; http://csrc.nist.gov/publications/PubsDrafts.html#SP-800-63-Rev.%201.
9. NIST. National Strategy for Trust Identities in Cyberspace (NSTIC); http://www.nist.gov/nstic/.
10. Rosen, B. and Polk, J. Best current practice for communications services in support of emergency calling; draft-ietf-ecrit-phonebcp-17 (work in progress), March 2011.
11. Rosen, B. et al. Framework for emergency calling using Internet multimedia; draft-ietf-ecrit-framework-12 (work in progress).
12. Taylor, T. et al. Security threats and requirements for emergency call marking and mapping. RFC 5069, January 2008.
13. Tschofenig, H., Schulzrinne, H., and Aboba, B. Trustworthy location information, draft-ietf-ecrit-trustworthy-location-02 (work in progress), May 2011.

Hannes Tschofenig (hannes.tschofenig@gmx.net) is employed by Nokia Siemens Networks in Finland and spends most of his time in the Internet Engineering Task Force (IETF) with the design of Internet protocols. He is a member of the Internet Architecture Board and co-chairs the IETF Web Authorization Protocol (oauth) working group. For many years he was co-chair of the IETF Emergency Context Resolution with Internet Technologies (ecrit) working group.

Calendar of Events

November 17–20
11th Koli Calling International Conference on Computing Education Research,
TBA, Finland,
Contact: Ari Korhonen,
Email: ari.korhonen@hut.fi

November 21–24
International Conference on Management of Emergent Digital Ecosystems,
San Francisco, CA,
Contact: William Grosky,
Email: wgrosky@umich.edu

November 28–December 1
ACM Multimedia Conference,
Scottsdale, AZ,
Contact: Kasim Selcuk Candan,
Email: candan@asu.edu
Phone: 480-965-2770

November 28–December 2
The Annual Meeting of the Australian Special Interest Group for Computer Human Interaction,
Canberra, Australia,
Contact: Duncan Stevenson,
Email: duncan.stevenson@anu.edu.au

December 3–7
The 44th Annual IEEE/ACM International Symposium on Microarchitecture
Porto Alegre, Brazil,
Sponsored: SIGMICRO,
Contact: Luigi Carro,
Email: carro@inf.ufrgs.br

December 4–5
Computer Human Interaction for the Management of Information Technology,
Cambridge, MA,
Sponsored: SIGCHI,
Contact: Adam S. Moskowitz,
Email: asm2acm@menlo.com

December 5–7
International ICST Conference on Bio-Inspired Models of Network, Information and Computing Systems,
York, United Kingdom,
Contact: Timmis Jonathan,
Email: jtimmis@cs.york.ac.uk

December 5–7
The 13th International Conference on Information Integration and Web-based Application & Services,
Hue City, Vietnam,
Contact: Rahayu Wenny,
Email: w.rahayu@latrobe.edu.au

DOI:10.1145/2018396.2018406

Scott Wallsten

Economic and Business Dimensions

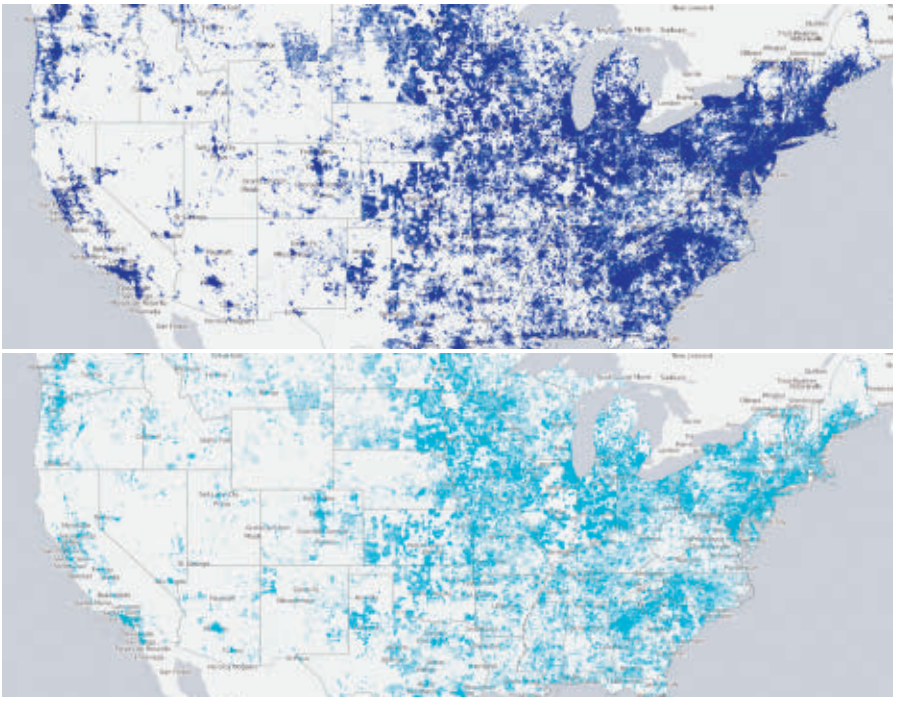
What Gets Measured Gets Done

Stop focusing on irrelevant broadband metrics.

"U.S. broadband is terrible" has become a familiar meme. An article in *Scientific American* last year fretted, "our creaky Internet makes it harder for U.S. entrepreneurs to compete in world markets."[4] Given the growing importance of broadband Internet connections to our work, civil society, and entertainment, a poor broadband infrastructure would indeed be cause for concern.

As it turns out, however, much of this concern is misplaced. It arises from a combination of the focusing on the wrong metrics, a misguided interpretation of consumer preferences, and a popular obsession with rankings. These misperceptions translate into misdirected, if well-intentioned, public policies that waste scarce resources.

Even worse, we do face real problems and issues with respect to broadband—a significant income-based digital divide, for example, and inefficient use of spectrum—but our singular focus on almost meaningless metrics and rankings distracts from more important issues.



Maximum advertised speed available (top) compared with type of technology available (bottom).

MAPS COURTESY OF HTTP://WWW.BROADBANDMAP.GOV

Adoption and Speed

The most commonly compared broadband metrics are adoption and speed. Conventional wisdom holds that both are too low in the U.S. While it is impossible to know what the right levels are, more careful analysis suggests that neither is a problem.

Adoption. Twice a year the Organization for Economic Cooperation and Development (OECD) reports that the U.S. ranks right around the middle of all OECD countries in the number of wired broadband connections per capita. That ranking, however, is increasingly meaningless in rich countries for the simple reason that multiple people in a household share each wired connection and average household sizes differ across countries. Countries with relatively large households, like the U.S. and Japan, are doomed to low per capita rankings. Consider that if every household in every OECD country had a wired broadband connection the U.S. would rank 17th or 18th on a per capita basis due to household size alone.

Moreover, broadband is available in the U.S. almost everywhere. Accord-

ing to the National Broadband Map, approximately 98% of U.S. households have Internet access with speeds at least 768kbps downstream and at least 200kbps upstream, and 96% have access to broadband of at least 3mbps downstream and 768kbps upstream. Nearly everyone without terrestrial access can purchase service from two satellite providers that are both in the process of significantly upgrading their service.

While availability is not a significant problem, a large income-based digital divide remains: poor people adopt broadband at substantially lower rates than wealthier people. Yet, U.S. policy does not focus on changing adoption. It focuses on building out to underserved areas, a less effective way to increase adoption.[2] For example, the $7.1 billion in broadband stimulus grants focused almost exclusively on building infrastructure, and the enabling legislation even barred the program from granting subsidies to individuals rather than firms. Current efforts to reform universal service suggest this focus is unlikely to change much.

Speed. Average advertised download speeds in many OECD countries are generally faster than they are in the U.S. As the accompanying figure indicates, however, the means of advertised plans do not reflect the speeds consumers actually purchase or receive. As it turns out, measured speeds are remarkably similar across rich countries.

Conventional wisdom holds that faster broadband speeds are always better, but is faster more useful? Most consumers do not value very high speeds and do not purchase those speeds even when they are available. It is true that speeds considered acceptable in the early days of DSL are too slow for many of today's common applications. But even today, speeds faster than approximately 10mbps deliver little incremental value for the simple reason that even the most bandwidth-intensive uses, like streaming high-definition video, require much less. Netflix and Amazon, for example, stream high-definition video at under 5mbps. A broadband connection cannot pull in the video faster than it is being pushed out.

Speed seems to be of so little concern to most U.S. broadband users

Conventional wisdom holds that faster broadband speeds are always better, but is faster more useful?

that 80% of them do not even bother to remember or check their own speed. A recent FCC survey found that 80% of U.S. broadband users did not know the speeds of their home broadband connections, yet 50% of users reported being "very satisfied" and 41% reporting being "somewhat satisfied" with their speed. In a detailed study of residential broadband demand in the U.S., Rosston, Savage, and Waldman[1] found that consumers were willing to pay about $80 per month for a reliable, "fast" connection, but were willing to pay only an additional $3 per month for a "very fast" connection.

To be sure, demand for speed will continue to change over time, as it has since the Bell 103 modem first communicated at a blazing 300bps, and someday we might consider today's speeds similarly absurdly slow, but no evidence suggests speeds are holding back innovation today. The typical purchased and available speeds in nearly every OECD country already exceed the bandwidth required for commonly used applications.

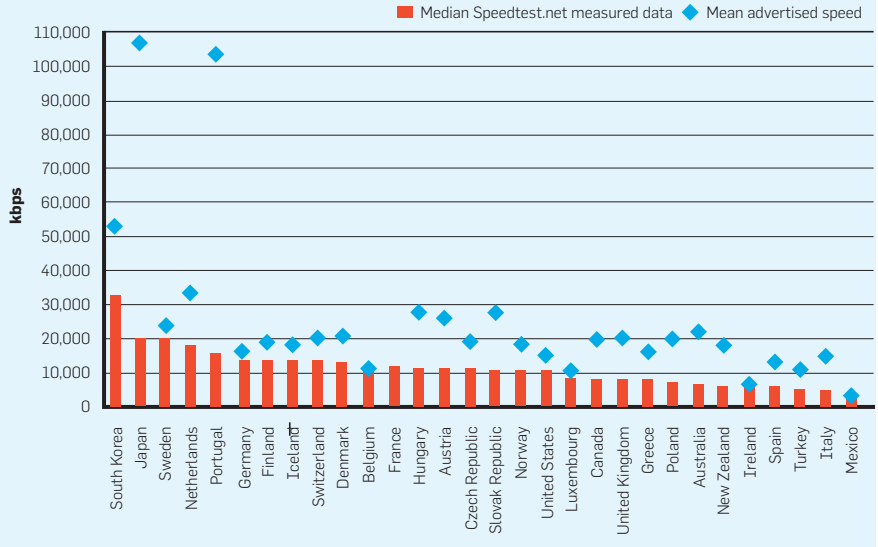
What Should We Measure?

Comparing performance across countries can be valuable, but we should focus on the right things.

Wireless Inputs and Outputs. A few years ago broadband implicitly meant wired connections. Within wired broadband, even as late as 2009 many industry observers thought the future of broadband exclusively meant fiber. Cable's DOCSIS 3.0 technology improved the capacity of cable broadband to such an extent that some analysts believe hybrid-fiber coaxial connections will allow the cable industry to dominate the wired market in much of the U.S.

Perhaps even that prediction is changing. The iPhone, iPad, Android operating system, and related app stores have made wireless an increasingly important part of the broadband ecosystem. Soaring wireless broad-

Measured and advertised download speeds (in kbps).



Source: Speedtest.net, OECD
Note: Speedtest.net: median "country daily speed" in Q2 2010 as calculated by the author from Net Index Source data.

Perhaps lack of speed is not a barrier to viable new applications while other aspects of quality are.

band use has implications for the direction of broadband innovation, competition, and adoption.

If the trend toward wireless use and mobility continues—and there is no guarantee it will, given the rapid succession of changes in what we think is important—then issues like spectrum policy should move to the forefront of all broadband policy issues. But we have little detailed cross-country information spectrum policy.

Business Use. Popular broadband metrics contain another misleading feature: they focus on residential broadband. Yet, residential broadband connections are unlikely to have large effects on net economic activity (see Wallsten[3]). Residential connections are used primarily for personal communication, shopping, and consuming news and entertainment. Much of business-to-consumer e-commerce, for example, reflects a shift in economic activity from "brick and mortar" to online retail rather than new economic activity. These activities largely represent transfers of economic activity rather than net new economic activity.

How digital communications technologies change business production processes, however, is more likely to determine whether these new technologies will have transformative economic effects. The direct economic effects of business use dwarf residential use. According to the U.S. Census, while business-to-consumer revenues reached almost $300 billion in 2009, they were an order of magnitude less than business-to-business revenues of about $3.1 trillion.

To be sure, productivity benefits may ultimately flow from residential broadband. Telecommuting, for example, could reduce resources society consumes, such as those used for physically commuting. That is only beginning to happen.

In short, how business incorporates digital communications technologies will have a much bigger effect on our standard of living over the next 20 years than will whether we reach 70% household broadband penetration in six months or a year.

Quality of Service Beyond Speed. Speed is but one element of broadband quality. Other factors like jitter, latency, and lack of fluctuations in quality also matter, but we know almost nothing about how consumers value other attributes of quality. Perhaps lack of speed is not a barrier to viable new applications while other aspects of quality are.

Conclusion

Focusing on the wrong metrics will do more harm than good. If we care about broadband adoption then we should stop focusing on availability. It is a much smaller problem. If we are worried about broadband quality, then we should focus on the aspects of quality businesses and consumers truly value, not merely speed. If we are worried about how broadband affects entrepreneurship and economic growth, then we should focus on barriers businesses face in integrating connectivity into their production processes. If we believe wireless connectivity is increasingly important, then we should focus on developing metrics for wireless and spectrum. C

References

1. Rosston, G.L., Savage, S., and Waldman, D. Household demand for broadband Internet service. *The B.E. Journal of Economic Analysis and Policy 10*, 1 (Sept. 9, 2010); http://www.bepress.com/bejeap/vol10/iss1/art79/.
2. Rosston, G.L. and Wallsten, S. The path to universal broadband: Why we should grant low-income subsidies and use experiments and auctions to determine the specifics. *The Economists' Voice 8*, 1 (Apr. 2011)
3. Wallsten, S. The future of digital communications and research. *Federal Communications Law Journal 63*, 1 (Dec. 2010), 33–42.
4. Why broadband service in the U.S. is so awful and one step that could change it. *Scientific American* (Oct. 2010).

Scott Wallsten (scott.wallsten@gmail.com) was the economics director for the FCC's National Broadband Plan. He is currently Vice President for Research and Senior Fellow at the Technology Policy Institute and a Senior Fellow at the Georgetown Center for Business and Public Policy.

DOI:10.1145/2018396.2018408

Pamela Samuelson

Legally Speaking
Why the Google Book Settlement Failed—and What Comes Next?

Assessing the implications of the Google Book Search settlement.

ON OCTOBER 28, 2008, Google, the Authors Guild, and the Association of American Publishers (AAP) announced a settlement of lawsuits charging Google with copyright infringement for scanning in-copyright books from the collections of major research libraries. While litigants can ordinarily settle lawsuits without judicial oversight, different rules apply in class action lawsuits. Because class action settlements affect the rights of many people who were not directly involved in the lawsuit or settlement negotiations, judges must determine whether the proposed settlement is "fair, reasonable, and adequate" to the class on whose behalf the lawsuit was being settled.

Just over 13 months after the fairness hearing on the proposed Google Book Search (GBS) settlement, Judge Denny Chin finally ruled that this agreement did not satisfy the fairness standard. The litigants did not appeal rejection of the settlement. The default next step is for the case to go back into litigation on the fair use or infringement issue.

In this column, I explain why Judge Chin disapproved the GBS settlement, speculate that the fair use issue may not be decided by the courts, and discuss the possibility of a new settlement and of legislation as alternatives.

PHOTOGRAPH BY MENNA / SHUTTERSTOCK.COM

"A Bridge Too Far"

The single most important factor in Judge Chin's ruling against the GBS settlement lay in his agreement with the U.S. Department of Justice (DOJ) that it was "a bridge too far."

The actual issue in litigation was whether scanning books to index their contents and provide snippets was copyright infringement or fair use. Yet, the settlement proposed an extremely complex forward-looking commercial regime under which Google could commercialize all out-of-print books (unless rights holders showed up to say no) and display up to 20% of contents of these books in response to search queries, as long as it shared 63% of the revenues with rights holders who registered with a new collecting society to be known as the Book Rights Registry (BRR).

Google never claimed it would be fair use to sell individual copies of out-of-print works to the public, nor to construct an institutional subscription database (ISD) of out-of-print books to

license to institutions of higher education, among others. Nor could it credibly make such a claim. Yet, the proposed GBS settlement would give it rights to do both of these things (and more).

The scope of the settlement, in other words, went far beyond the issue in litigation. At the fairness hearing the DOJ lawyer pointed out it was the duty of class counsel to litigate the claims they brought or to settle those claims. They had instead used the existence of a dispute about GBS scanning to remake the market for e-books and change the default rules of copyright law (which generally require a prospective user to get permission in advance before making commercial uses of the works), as the proposed GBS settlement would arguably do.

Litigation or Legislation?

Judge Chin also agreed with the DOJ that the only legitimate way to restructure rights and e-book markets in the manner proposed in the GBS settlement was through legislation.

The quasi-legislative character of the settlement was most evident in its solution to the so-called "orphan works" problem. Works are deemed orphans when their rights holders cannot be found through a reasonably diligent search. A book published in 1953, for instance, may still be in copyright whose owner is a firm that no longer exists and/or an author who died without heirs. In 2006, the U.S. Copyright Office proposed legislation to allow orphan works to be made more accessible, but so far this legislation has not been enacted by Congress. (The EU has recently proposed a directive addressing the orphan works problem.)

The proposed settlement would have given Google the right to commercially exploit all orphan books because their rights holders were members of the class that would have virtually consented to these uses through the judge's approval of the settlement. Judge Chin decided it was for Congress, not the courts, to address the orphan works problem.

Adequacy of Representation

The *Authors Guild* complaint named three of its member authors as representatives of the class of authors affected by GBS scanning. These authors, Guild lawyers, and lawyers designated as counsel for the class have an obligation to represent the interests of all members of the class.

In my submissions to the court, I argued that the plaintiffs and their lawyers had not adequately represented the interests of academic authors. Unlike Guild authors, academic authors would be inclined to think that scanning books to index them was fair use, not copyright infringement. They would, moreover, be likely to want their out-of-print books to be available on an open access basis rather than through a profit-maximizing scheme such as the GBS settlement proposed. Judge Chin agreed with me that academic authors had different interests than Guild authors and that the Guild's lawyers had not adequately represented our interests.

Judge Chin was also plainly affected by the large outpouring of opposition to the GBS settlement from other copyright owners. He noted that 6,800 authors had opted out of the settlement because they did not wish to be bound by it. He quoted at length from author objections to GBS and to the settlement. The governments of France and Germany and many foreign rights holders also opposed it. Although not ruling on contentions that the proposed settlement violated U.S. treaty obligations, Judge Chin made it clear he was troubled by these assertions.

Although saying that it was not a ground for disapproval of the settlement, Judge Chin also expressed concern about the lack of user privacy protections. The GBS settlement called for extensive collection of data about individual reader uses of GBS books, but had virtually no provisions limiting what Google could do with this information. In addition, he expressed reservations about the antitrust concerns raised by the DOJ and by Yahoo! and Microsoft about the extra advantage that Google would have in the search market by getting a license to improve its search engine with GBS books.

The actual issue in litigation was whether scanning books to index their contents and provide snippets was copyright infringement or fair use.

Benefits of the Proposed Settlement

Controversial as it was in some respects, the GBS settlement, if approved, would have brought about many socially beneficial results. Chief among them was a vast expansion of access to out-of-print but in-copyright books. Up to 20% of their contents could have been displayed to users in response to search queries. Millions of these books would have been freely accessible at terminals at public libraries (one per library) and at institutions of higher education (one per so many students), as well as through institutional subscriptions available to libraries and other institutions. E-book versions of these out-of-print books would also have been available for purchase by consumers, which they could access "in the cloud." In addition, Google pledged in the proposed settlement to make digitized copies of these books available in formats accessible to print-disabled persons (such as versions in Braille or with enlarged typography).

There are already more than 15 million books in the GBS corpus, the overwhelming majority of which come from the collections of major research libraries. These collections are dense with the accumulated knowledge of the ages, and Google is scanning more of them every day. It was thus no exaggeration to assert that approval of the settlement would have vastly expanded access to our cultural heritage.

The GBS settlement would have permitted Google to provide its library partners with copies of scans of books from their collections that could be used for preservation purposes and for "non-consumptive research" (for example, tracing the influence of a thinker over time or the origins of words). Google itself would have been privileged by the settlement to engage

in non-display (computational) uses of books in the GBS corpus for purposes such as improving its search technologies and automated translation tools.

The proposed settlement would also have been socially beneficial in providing new income streams to authors and publishers through the BRR. These benefits cannot be realized through a class action settlement, but can they be achieved in other ways?

Controversial as it was in some respects, the GBS settlement, if approved, would have brought about many socially beneficial results.

What's Next?

Judge Chin made clear that he would look more favorably on an opt-in settlement (that is, requiring Google to get permission from rights holders before commercializing their books) than he had on the opt-out settlement proposed in 2008. However, lawyers for Google and the Authors Guild have told the judge that Google has no interest in an opt-in settlement. An opt-in settlement would also not bring about the socially beneficial results envisioned in the proposed GBS settlement. Yet, because litigation is very expensive, takes a long time, and poses risks for both sides, settlement is far more likely than resuming litigation at this point, although Judge Chin has indicated that because the parties have not yet reached a new settlement, the case will be scheduled for trial next July.

Legislation would be another way to accomplish some of the socially beneficial aspects of the GBS settlement. Maria Pallante, the newly appointed Register of Copyrights, and James Billington, the Librarian of Congress, have written to key Congressional leaders to indicate their willingness to undertake a study of legislative options in the aftermath of the GBS settlement disapproval.

Having studied the settlement and assessed its possible benefits, I have developed a framework for a legislative proposal that would aim to achieve these objectives.[1] In brief, I recommend: 1) creating a privilege to scan in-copyright works for preservation purposes, to allow their contents to be indexed, and to allow non-display uses of the scans, including non-consumptive research uses; 2) allowing "orphan works" (works whose rights holders cannot be found after a reasonably diligent search) to be made available on an open access basis; 3) expanding the right of libraries and others to improve access for print-disabled persons; and 4) ensuring that reader privacy interests are respected. Unfortunately, the political economy of copyright in the U.S. does not bode well for these proposals.

I also suggest that consideration be given to creating an extended collective licensing regime for out-of-print, non-orphan books so that an ISD such as the GBS settlement proposed might be created. Extended collective licenses have been used with considerable success in Nordic countries to provide rights holders with compensation while at the same time allowing users the assurance they can get a license to make a large number of works available even when transaction costs of clearing all rights, one by one, would be excessive or possibly prohibitive.

Many, even if not all, of the social benefits that would have flowed from approval of the GBS settlement can be achieved in other ways. Some reforms can be done through private ordering (for example, professors making their books available on an open access basis), some through fair use (for example, scanning to index contents), and some through legislation. We should not let the failure of the GBS settlement stand in the way of finding new ways to make cultural heritage more widely available. C

Reference

1. Samuelson, P. Legislative alternatives to the Google Book Settlement. *Columbia Journal of Law & Arts* (forthcoming 2011); http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1818126

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

DOI:10.1145/2018396.2018407

Michael Davis

Computing Ethics
Will Software Engineering Ever Be Engineering?

Considering whether software engineering and engineering can share a profession.

An anonymously attributed adage states: "With another name, social engineering would not be mistaken for engineering." Approximately 15 years ago, I published a short article in the *Journal of Engineering Education* arguing—among other things—that software engineering was not then engineering.[1] I have now been asked whether enough has changed to make me think software engineering is engineering. My answer is: much has changed—with some changes weakening the separation between engineering and software engineering and some reinforcing it—but, overall, the argument stands. This answer will surprise those who, unaware of that article, think software engineering's status as engineering is obvious. I therefore think it wise to precede any explanation of why software engineering is not engineering by disposing of a few unexamined presumptions that might make software engineering's status as engineering seem obvious.

The Software Engineering Code of Ethics and Professional Practice differs in significant ways from all the engineering codes I know.

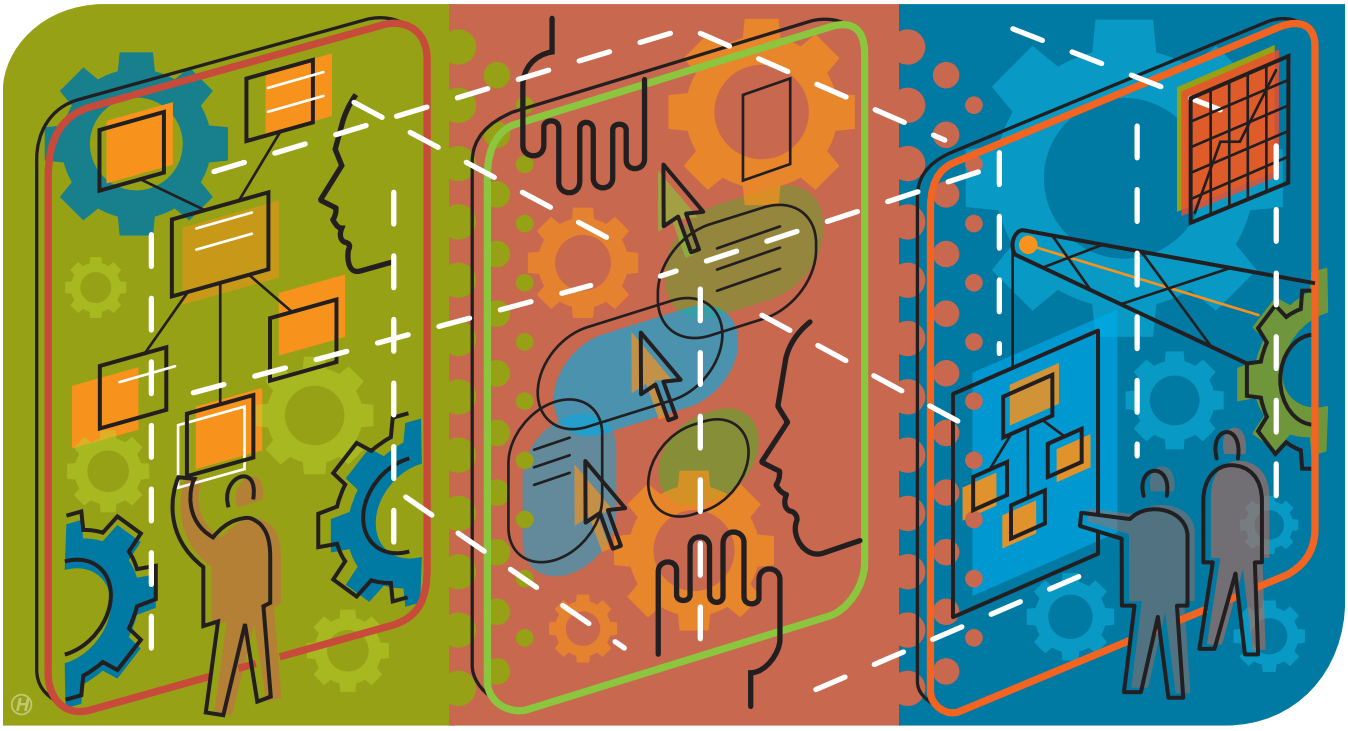
Senses of Engineering

"Engineering" has at least four senses in English. One, the oldest, understands engineering as tending engines (originally, "engines of war"). Casey Jones was an engineer in this sense; so is the custodian of my building, a licensed "boiler engineer"; and so too, the sailor rated "marine engineer." Neither engineers (strictly speaking) nor software engineers are engineers in this sense.

Almost the opposite of this first sense is what we might call the functional sense, engineering-as-invention-of-useful-objects. In this sense, the first engineer may have been the caveman (or cavewoman) who invented the club, cutting stone, or fire pit. Though this sense would certainly make software engineers engineers, there are at least two reasons to reject it here. First, the functional sense is too broad. Architects, industrial designers, and even weekend inventors are all engineers in this sense, making software engineering's claim to be engineering uninteresting. Second, the functional sense is anachronistic. It takes a sense of "engineering" that did not exist much before 1700 and applies it to cavemen, carpenters, tinkerers, and the like, who would have understood themselves quite differently.

The functional sense of engineering nonetheless seems relevant here. Software engineering's official Body of Knowledge offers this definition of software engineering: "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, and the study of these approaches; that is, the application of engineering to software."[2] The Body of Knowledge assumes, without argument (a mere "that is"), that engineering is a certain function, any "systematic, disciplined, quantifiable approach to the development, operation, and maintenance [of something]". That assumption must be false. It would force us, for example, to rank accounting—a field no one supposes to be engineering—as "financial-records engineering" (since accounting is a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of financial records).

Closer to our subject is a third sense, engineering-as-discipline. A discipline is a distinctive way of carrying on an activity, some combination of knowledge, skill, and judgment that must be



learned. Any craft or trade has its discipline—as do many activities that are not craft or trade, such as meditation or calisthenics. In this sense, neither architects, nor industrial designers, nor weekend inventors are engineers. Architecture and industrial design each have a discipline easily distinguished from engineering's. Weekend inventors have no discipline at all; they may invent any way they like.

Software engineering is not engineering in this third sense. The body of knowledge engineers are supposed to learn differs in important ways from software engineering's body of knowledge. So, for example, engineers have to take courses concerned with the material world, such as chemistry and statistics; software engineers do not. Software engineering's official Body of Knowledge was in fact an important step in clarifying the distinction between engineering proper and software engineering. It requires software engineers to know things other engineers do not and not to know some things other engineers do know.

Software engineering has, indeed, become a profession. What it has not become is part of the engineering profession.

The last sense of engineering we need to distinguish here is engineering-as-profession. A profession is (we may say) a number of individuals in the same occupation voluntarily organized to earn a living by openly serving a moral ideal in a morally permissible way beyond what law, market, morality, and public opinion would otherwise require.[a] An occupation is a discipline by which one may, and some do, earn a living. Both engineering and software engineering are now occupations but, having (as just noted) different disciplines, must be different occupations. That is one reason why they cannot share a profession. There is another.

The Software Engineering Code of Ethics and Professional Practice differs in significant ways from all the engineering codes I know. Software engineers are, for example, supposed to "[m]oderate the interests of the software engineer, the employer, the client and the users with the public good" (1.02).[b] Engineers do not now have such a duty to moderate.

Software engineering has, indeed, become a profession. What it has not become is part of the engineering profession. Anyone who claims otherwise must find a sense of engineering different from those distinguished here, one that makes software engineering a part of engineering without including as well disciplines, occupations, or professions, such as architecture or accounting, that clearly are not part of engineering.

Professions are voluntary associations. You cannot become a member simply by claiming to be one. You must be admitted (by the profession, not just by a technical society like the ACM). Engineering has a long history of other occupations claiming to be engineering: recent examples include genetic engineering (a kind of tinkering with genes); reengineering (a fad in management); and financial engineering (gambling on Wall Street). Software engineering actually began with an attempt to copy engineering practices, making its claim to be engineering more respectable

a For a defense of this definition, see my article "Is Engineering a Profession Everywhere?" *Philosophia 37* (June 2009), 211–225.

b Software Engineering Code of Ethics and Professional Practice (1999); http://www.acm.org/about/se-code. For history of this document, see my essay "Code Writing: How Software Engineering Became a Profession," Center for the Study of Ethics in the Professions, Chicago, 2007; http://hum.iit.edu:8080/aire/sea/1/book/index.html.

ILLUSTRATION BY HANK OSUNA

than most. But the enormous complexity of software has forced software engineering to develop in ways engineering has not—and may never.[c] Many of the very methods that make software engineering useful distinguish it from engineering. Engineers have good reason to continue to treat software engineers as belonging to another profession.[d]

I have, I hope, just explained why I still think software engineering is not engineering in a way that engineers should recognize. I now want to point out four reasons to think that engineering might someday merge with software engineering. All four are, oddly, changes in engineering, not software engineering.

▸ Electrical and computer engineering (ECE) is often thought to be the field of engineering closest to software engineering. Over the last decade, ECE has become less committed to traditional engineering courses concerned with the material world. So, for example, a number of ECE departments, including the one at the University of Illinois at Urbana-Champaign, have stopped requiring statics, dynamics, and thermodynamics. If that trend continues, then either ECE will split off from the main body of engineering or engineering's core of required *engineering* courses will increasingly resemble software engineering's.

▸ Since the 1700s, engineers have had to know just two natural sciences: physics and chemistry. Recently, some programs in environmental engineering, biomedical engineering, and agricultural engineering have begun to allow students to substitute biology for physics or chemistry. For engineers, this makes sense, since several of the new frontiers of engineering rely on biology rather than physics and chemistry (as until recently). But, if this trend continues, engineering's *science* core will increasingly resemble the science courses software engineers take to satisfy general distribution requirements.

Whether or not software engineers do any engineering, engineers increasingly engage in activities that look like software engineering.

▸ Engineers are increasingly replacing mechanical systems with software. Not only do most engineers now use software regularly, many write specifications for software, modify existing programs themselves, or even write (simple) programs. Whether or not software engineers do any engineering, engineers increasingly engage in activities that look like software engineering (even if these engineers do not call themselves "software engineers" and do not work the way that software engineers would). Whether some fields of engineering will dissolve into software engineering seems an open question.

▸ Computer science used to have an accreditation body separate from engineering's. That is no longer true. All computer science programs, including software engineering, are now under engineering's accreditation body, ABET. Of course, the accreditation process and standards distinguish between engineering programs and computer science programs. But that distinction does not preclude eventual merger. ABET has always distinguished between various fields (or subdisciplines) of engineering. So, for example, it always sent mechanical engineers to review a mechanical engineering program; electrical engineers, to review an electrical engineering program; and so on. The expansion of ABET's accreditation powers makes it easier than before for software engineering to merge into engineering, indeed, for all of computer science to do that.

Having pointed out four reasons that seem to point to software engineering's eventual merger with engineering, I now point out three reasons to believe the merger will not happen soon, if at all:

▸ All engineering is still fundamentally about physical systems; software engineering is not. Even a field so closely allied to software engineering as computer engineering must take into account physical factors in design, for example, heat produced in a microchip or speed of electrical current, to a degree software engineers do not.

▸ Software engineering is today a large profession, indeed, one of the largest—half the size of engineering, true, but about the same size as medicine or law. With so many practitioners, software engineering is more likely to divide than to join up with another large profession.

▸ If computer science ever ceased to be the home of software engineering, the most likely new home might well be management information systems or information technology management. These business disciplines resemble software engineering at least as much as engineering does. In practice, most software engineers work more with information systems managers than with engineers.

Conclusion

Whether knowledge of the future is possible is a perennial question in philosophy. What is certain is that prophets are seldom right on any important question. So, I make no claim to know whether software engineering will ever merge with engineering. I claim only to know that—despite the common term "engineering"—software engineering is not now engineering. C

c Michal Young and Stuart Faulk, "Sharing What We Know About Software Engineering," in *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (FoSER '10), ACM, 439–442, argue that engineering has much to learn from software engineering—inadvertently making clear how much engineering's discipline differs from software engineering's.

d For a darker route to this conclusion, see David L. Parnas, "Risks of undisciplined development," *Commun. ACM 53*, 10 (Oct. 2010), 25–27. Note that Parnas, though a star of software engineering, is an electrical engineer—both by discipline and declaration—looking at software engineering the way knowledgeable engineers typically do.

References

1. Abran, A. et al. *Guide to the Software Engineering Body of Knowledge—2004 Version*. IEEE Computer Society (2004), 1.
2. Davis, M. Defining engineering: How to do it and why it matters. *Journal of Engineering Education 85*, (Apr. 1996), 97–101.

Michael Davis (davis2643@gmail.com) is a senior fellow at the Center for the Study of Ethics in the Professions and a professor of philosophy at Illinois Institute of Technology, Chicago.

Thanks to Keith Miller and Rachelle Hollander for helping me think through this column.

DOI:10.1145/2018396.2018409

SIGCSE Teaching-Oriented Faculty Working Group

Education

Teaching-Oriented Faculty at Research Universities

Developing a well-rounded university through specialization.

Titles and job details vary from university to university, but teaching-oriented faculty (TOF) contribute significantly to the mission of CS departments in U.S. and Canadian research universities. The nine collaborators of this column are TOF in computer science departments at research universities in the U.S. or Canada; some of us even have the word "Professor" in our job titles. Here, we describe how positions like ours work, how they contribute to education—and as a side effect, to research—at our institutions, and how departmental policies can influence TOF's success and satisfaction.

Ever since I taught my first undergraduate class, when I was a graduate student, many of my students have called me "Professor." I wasn't a professor then, and I'm not one now, even though I have a tenured faculty position at a research university. My title is "Senior Instructor" and I'm one of a growing category of faculty in research universities CS departments, faculty whose primary focus is teaching.

Steve Wolfman

-SIGCSE TOF WORKING GROUP MEMBER

Excellent Teachers Who Prioritize Teaching

The unifying characteristic of TOF is excellent teaching. We teach large classes, introductory classes, specialized classes; we typically teach more classes and more students than other faculty in our departments and by common measures we do it very well. Unlike our non-TOF colleagues, we are also evaluated primarily on the basis of our teaching.

TOF are not hired as inexpensive, one-shot instructors to fill temporary gaps in the course list. Most of our institutions have different job titles for this kind of faculty appointment. Nor are TOF positions intended for traditional faculty who are no longer active in research. TOF have a primary profes-

ILLUSTRATION BY MIA ANGELICA BALAQUIOT

sional focus on teaching computer science at the undergraduate level and a passion for excellence in CS education.

At the University of British Columbia (UBC), for example, many CS faculty are excellent teachers but only TOF are hired and promoted primarily as educators. A TOF member teaches twice as many courses as a non-TOF member and a disproportionate share of the undergraduate students. Over the last three years, nine TOF taught 47% of the undergraduate enrollment while 42 non-TOF taught 42% (and other temporary instructors taught 11%). How well do the TOF teach? UBC's campuswide teaching award is a plausible, succinct measure of teaching excellence, awarded on a broad basis including peer and student observations, student evaluations, and a teaching portfolio. Five of the 10 current TOF and three of the 49 current non-TOF have received the award.

Judicious use of TOF furthers the university's teaching mission by enabling some specialization. A department can favor assignment of TOF to courses—often those with large and academically diverse student populations—that demand particular effort and attention from experienced, committed faculty with a strong focus on teaching. Likewise, a department can favor assignment of research-oriented faculty to courses—likely advanced—in which their research naturally connects to instruction. (In the UBC example mentioned previously, 65% of the undergraduate courses taught by non-TOF were upper-division electives.)

A common criticism of such specialization is that all good university teaching requires active research. This is an attractive idea for research universities balancing research and teaching missions and their stakeholders. However, research on university teachers has established that active research is not a predictor of effective teaching.[1] On the other hand, TOF do indirectly support the university's research mission by allowing non-TOF to take a lighter teaching load with a tighter connection to their research.

...And Something Else

TOF have impact outside of the classroom as well. TOF jobs typically involve teaching *and something else* that affects undergraduate education. For example, the University of California (UC), Berkeley hired its first TOF member in CS not just to teach but also to radically restructure the introductory CS curriculum. Indeed, all of the authors engage in professional activity beyond teaching, even where our departments impose no official requirement to do so.

TOF have a primary professional focus on teaching computer science at the undergraduate level and a passion for excellence in CS education.

We briefly describe some key types of "something else" contributions TOF make here. The majority of our institutions have TOF engaged in most of the contribution types we list, but for brevity and concreteness we illustrate each type with a few specific examples.

Curriculum Development. Introductory curriculum development is a common task for TOF. As with the Berkeley example earlier, UC Santa Barbara (UCSB) TOF contributed substantially to a lower-division curriculum redesign, integrating recent research in computer science pedagogy such as pair programming and foreshadowing of advanced research concepts to improve retention. At Princeton, TOF co-developed and teach an interdisciplinary introduction to CS taken by almost half of Princeton undergraduates. Such a large service course would not be possible without the focused support of TOF.

TOF often contribute in specialized areas as well. At UC Irvine, a TOF member developed and teaches a year-long capstone project course, recruiting project clients from industry. Another TOF member developed and taught a computer game development course and later co-designed a new "Computer Game Science" major.

Undergraduate Advising. TOF's extensive contact with undergraduates makes them natural mentors for the students. UCSB's two TOF piloted the faculty undergraduate advising program there, developed an undergraduate research course, provided undergraduate research opportunities, and brought undergraduates onto department committees. At UBC, TOF created and maintain a second degree program and a broad array of popular combined majors.

Research. Many TOF maintain research programs, especially research on CS education. At UBC, TOF have published computer science education research in SIGCSE and similar venues. They also supervise many undergraduate research assistants and sometimes co-supervise graduate students in both CS education and more traditional CS research. UCSB TOF have advised 11 undergraduate research assistants in the past three years and published during that time (often with the students) in top-tier architecture conferences and at SIGCSE.

Textbook and e-Course Authorship. Authorship of course materials is a common contribution of TOF. At Princeton, a TOF member co-authored a CS 1 textbook and updated a best-selling CS 2 textbook. A TOF member also curates two freely accessible book sites—with code, data sets, algorithm visualizations, assignments, and exercises—that receive half a million hits per month. One UCSB TOF member contributed to the past two editions of the top-selling computer architecture textbook. Similarly, TOF often design pedagogical software, such as the aforementioned textbook-support software and the WebCT course management system created by a TOF member at UBC.

K–12 Outreach. TOF have a special interest in how we recruit undergraduate CS students. At the University of Texas at Austin (UT), TOF created and sustain long-running K–12 outreach programs including a summer camp for high school girls—with 20% of 2010 participants now majoring in CS at UT—and an annual CS4HS program for 20–30 high school teachers. UT TOF also support high school computer science contests (similar to the ACM programming contest) for several thousand students each year. UCSB TOF recently developed and offered a summer camp with 40 students—most members of minority groups underrepresented in

computer science—30% of whom are now considering CS as a career.

University and Professional Service. As with other faculty, TOF often perform significant university and professional service. A UC Irvine TOF member recently chaired the UC-wide faculty senate committee on educational policy, helping develop the position of the faculty on issues such as the quality of online education, proposed measures to deal with budget cuts, and course transfers from community colleges. A TOF member at UBC has served for many years as Associate Dean for Curriculum and Learning. Another recently served as Program and then General chair for SIGCSE, ACM's major annual Computer Science Education symposium.

External Support. Many TOF secure external funding for projects like those listed here, including several of the specific projects described in this column. In addition, three Duke TOF have acted as principal investigators on grants from the National Science Foundation (NSF); received gifts from corporate sources to further education; and participated in multi-institution grants. One Berkeley TOF member's collaboration with an education researcher led to several NSF- and industry-funded projects.

Best Practices for Hiring, Retaining, and Supporting TOF

We have illustrated ways that talented, dedicated, and expert TOF can contribute to the mission of a research university. How then do research universities recruit the best talent, maintain their dedication, and retain them as they develop their expertise?

We believe the most successful TOF positions—measured from both TOF and non-TOF perspectives—are those viewed as approximately equivalent to other faculty positions but with a different focus. Problems occur when the position, the person hired as TOF, or the job responsibilities create a divide between the TOF member and their colleagues. Departments and institutions can "tune up" their TOF positions by moving toward practices that encourage a positive relationship.

How do research universities recruit the most talented TOF, maintain their dedication, and retain them as they develop their expertise?

Faced with a general shortage of educated workers in computing, teaching-oriented faculty can be an essential part of a CS department's solution.

For example, in hiring, favor full-time appointments and treat the hiring process with similar care and deliberation to that used for non-TOF positions. To support new TOF, provide mentorship, gradual ramp-up of duties, and regular feedback from peer observations. When evaluating TOF for promotion and merit, create a real career path including increased security (for example, through longer-term appointments) and recognize both teaching and the "something else" in reviews. To retain and support TOF in the long run, keep teaching loads sustainable, accounting for labor-intensive activities like managing the teaching staff of a large course, and provide TOF with the logistical resources they need given their high teaching loads and limited access to research funds. To integrate TOF into a department, grant and encourage full rights to participate in university affairs (for example, department meetings, committee membership, voting rights, and PI eligibility), emphasize the importance and challenge of effective teaching to all faculty, encourage TOF to participate (at a manageable level) in research groups, and encourage non-TOF to participate (also at a manageable level) in the aspects of undergraduate education primarily controlled by TOF.

Departments should also exploit their TOF's expertise—intense practical experience and (often) currency in general or disciplinary educational literature—by encouraging dissemination and use of pedagogical best practices among their faculty. The advice provided in this column to foster TOF will facilitate dissemination by marking both TOF and undergraduate education as valued by the institution. That status legitimizes formal discussions and "hallway chat" about pedagogical problems. Of course, departments should encourage all faculty (not just TOF) to excel in teaching and the study of learning and to contribute that expertise to these conversations.

Some of these practices may require changes to or novel interpretations of existing policy, but CS departments at many excellent schools have adopted them. To the extent official change cannot be achieved, a department can still create a strong and public cultural commitment to these practices and to its teaching-oriented faculty.

Conclusion

In the ultra-specialized world of the research university, it should come as no surprise that specialization in teaching serves the university's mission. University administrators benefit from increased stability and quality in undergraduate education and, indirectly, in research. Traditional faculty benefit from reduced and specialized teaching loads. Most importantly, students benefit from studying under a mixture of specialists, experts in groundbreaking areas of research but also experts in teaching. Faced with a general shortage of educated workers in computing, teaching-oriented faculty can be an essential part of a CS department's solution. C

Reference

1. Prince, M.J., Felder, R.M., and Brent, R. Does faculty research improve undergraduate teaching? An analysis of existing and potential synergies. *Journal of Engineering Education 96*, 4 (Apr. 2007), 283–294; http://www4.ncsu.edu/unity/lockers/users/f/felder/public/Papers/Teaching-Research(JEE).pdf

This column is a collaborative effort of teaching faculty at several institutions, with editorial contributions by Steve Wolfman, Owen Astrachan, Mike Clancy, Kurt Eiselt, Jeffrey Forbes, Diana Franklin, David Kay, Mike Scott, and Kevin Wayne.

DOI:10.1145/2018396.2018410

Douglas Baumann, Susanne Hambrusch, and Jennifer Neville

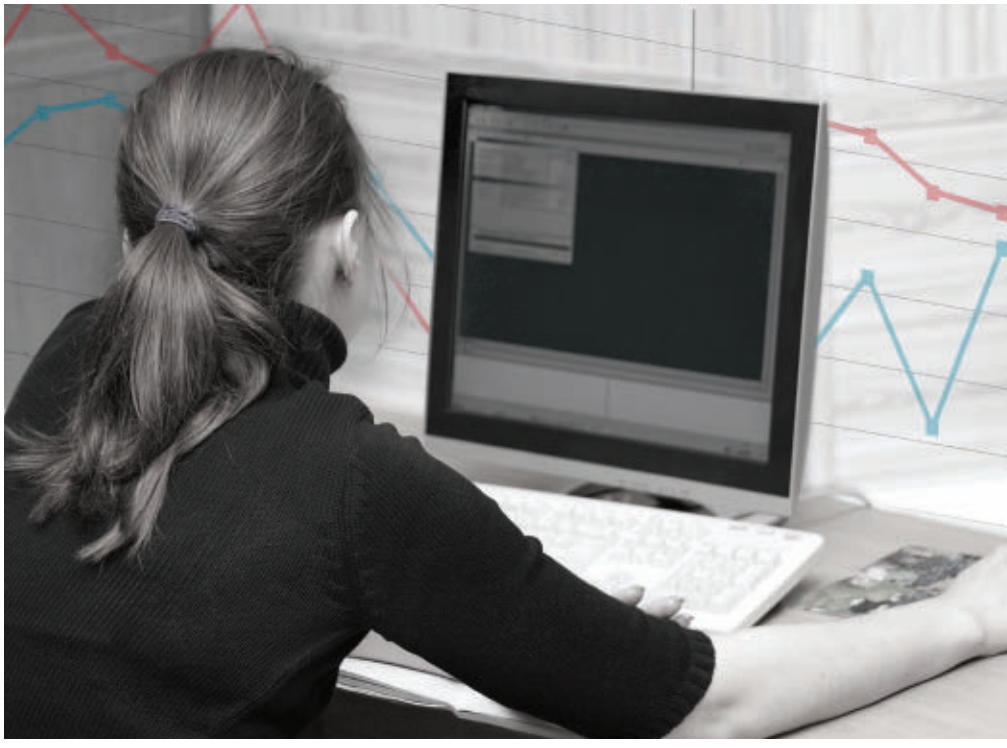
Viewpoint
Gender Demographics Trends and Changes in U.S. CS Departments

Using the past 10 years of Taulbee Survey data to evaluate female student enrollment across varied academic institutions and departments.

Ph.D.-granting departments in the U.S. claim to have increased the number of female faculty, to be graduating more Ph.D. students, and to have again increased their undergraduate enrollment, especially the enrollment of female students. This Viewpoint is derived from a report that used Taulbee Survey data from 1999–2009 to provide insight into these claims and other questions regarding gender demographics. In particular, we take a close look at whether the trends are different for private and public institutions, for large and small departments, and for departments of different ranks.

The data indicates that overall both the number and proportion of female tenured and tenure-track faculty as well as females receiving a Ph.D. have increased; however, these increases are not uniform across types of departments. In many highly ranked departments female full professors outnumber female faculty of other ranks and increases have been small. Departments that have significantly increased the number of female faculty tend to have done so by hiring female assistant professors. Although the overall Ph.D. production has doubled from 2002–2008, the proportion of Ph.D.'s awarded to females is lower in higher ranked departments. In contrast, higher ranked private institutions have a higher proportion of female undergraduates. This Viewpoint highlights some of these results and trends generated from Taulbee data; the full report is available at http://www.cs.purdue.edu/homes/seh/Taulbee99-09_Full.pdf.

The Taulbee Survey, conducted annually by the Computing Research Association (CRA), provides information on Ph.D., M.S., and B.S. enrollment and production in computer science, computer engineering, and information schools in Ph.D.-granting departments in North America. Results of the Taulbee Survey since 1992 are available at http://www.cra.org/resources/taulbee/. The Taulbee Survey uses the 1995 National Research Council (NRC) rankings to form four groups: The 36 top-ranked departments are partitioned by their rank into three



Due to the increase in the overall number and proportion, the number of females receiving a Ph.D. has more than doubled since 2002.

groups, 1–12, 13–24, and 25–36. The remaining departments belonging to the CRA, some ranked and some unranked by the NRC, form one group (referred to as 37+).

To refine the 127 departments in Taulbee's group "37+" and to provide insight into trends of public versus private institutions, our study uses a different partitioning of the 163 U.S. CS and CSE Ph.D.-granting departments. Groups had to be large enough to ensure no department could be identified and we wanted to retain some ability to compare our groupings against Taulbee Survey results. The underlying eight groups are formed as follows.

▸ The 36 top-ranked institutions are partitioned into four groups. They are first partitioned into public and private institutions (19 and 17 institutions, respectively). The 1995 NRC rankings are used to partition the private institutions into two groups by rank.

▸ The 127 institutions in group 37+ are also partitioned into four groups. The 31 private institutions form one group. The 96 public institutions are partitioned into three groups using their reported faculty size in 2007: large departments are those having more than 21 tenured and tenure-track faculty, medium departments have between 21 and 15 faculty, and small departments have fewer than 15 faculty.

The accompanying table shows the number of departments in each of the eight groups and the average response rate to the Taulbee Survey. A listing of the departments in each group is available in the full report.

Trends in Faculty Demographics

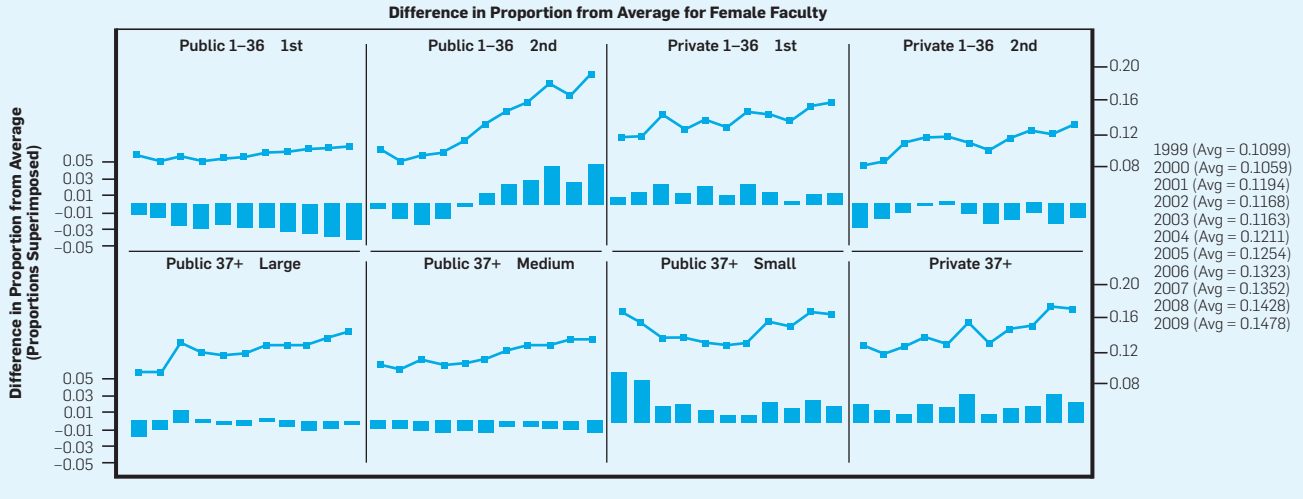
Since 1999, the number of faculty in CS and CSE departments in the U.S. has increased. For the 163 departments considered, the average number of faculty increased from 22 in 1999 to 28 in 2009. The number of female tenure-track and tenured faculty has also increased. In proportion, it increased from 11% to 14.8%. Looking at the eight groups, the average numbers for female faculty range from a minimum of 1.13 (for departments in group Public 37+ Medium in 2001) to a maximum of 7.57 (for Public 1–36 2nd in 2009). The next two figures provide insight into the increases for female faculty for the eight groups, one for proportions and one on average numbers for female faculty in the three academic ranks.

Figure 1 shows that each group has seen an increase in proportion in fe-

Group sizes and response rates.

Group	Number of Departments	Response Rate
Public Ranks 1–36 (1st)	10	100%
Public Ranks 1–36 (2nd)	9	84%
Private Ranks 1–36 (1st)	8	95%
Private Ranks 1–36 (2nd)	9	98%
Public Ranks 37+ (Large)	36	84%
Public Ranks 37+ (Medium)	26	92%
Public Ranks 37+ (Small)	34	74%
Private Ranks 37+	31	73%
	163	**87%**

Figure 1. Proportion of female faculty tenured and tenure-track faculty in the eight groups (lines) and the difference from the overall average across all departments (bars).



male faculty. Each panel shows a line representing the proportion of female faculty and bars indicating the difference from the overall average for the year. The average proportion of female faculty of 163 institutions forms the "baseline" of each year. The eight panels show that increases are not uniformly distributed across groups.

▸ Departments in *private* institutions ranked 1–36 show increases in the proportions but the relationship to the baseline average shows little change.

▸ Departments in *public* institutions ranked 1–36 show different trends: the departments in Public 1–36 1st saw almost no change in proportion, which results in a decrease from the baseline average; Public 1–36 2nd saw a significant increase from 10% in 1999, which was below the baseline average, to 19%, the highest among all groups in 2009.

▸ The small number of faculty in departments of Public 37+ Small make it difficult to interpret the trend in a meaningful way. For all three other panels for 37+ departments, an upward trend can be seen in the proportions, while the relationship to the baseline changed little.

In order to better understand increases in proportion and increases in numbers, we examined the average number of female faculty in each rank. Figure 2 shows the average number of female faculty in each group, partitioning the total number of female faculty into assistant, associate, and full professors. As Figure 1 shows, group Public 1–36 2nd has seen the largest proportional

Figure 2. Average number of female faculty in a department by academic rank.

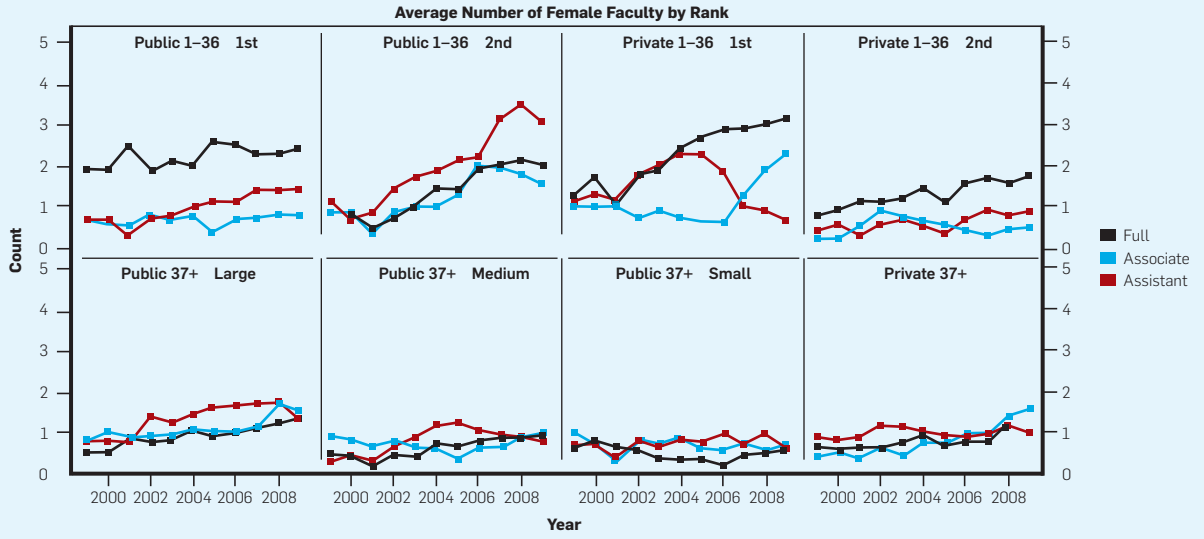
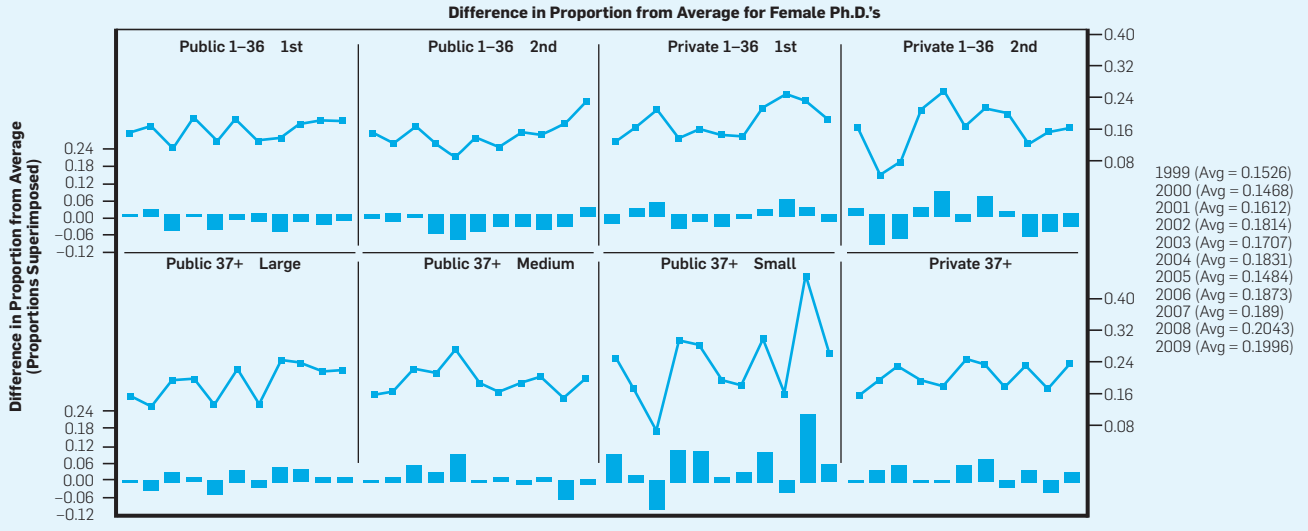


Figure 3. Proportion of Ph.D.'s awarded to females in the eight groups (lines) and difference in proportion from yearly average (bars).



increase in female faculty. From Figure 2 one can conclude that much of the increase is the result of hiring female assistant professors. Group Public 1–36 2nd is the only one of the four groups of rank 1–36 for which assistant professors outnumber the other ranks. Somewhat surprisingly, for all three other panels representing institutions of rank 1–36, the average number of female full professors outnumbers the other ranks. In the panels representing institutions of rank 37+, differences in the averages for different rank are small, with assistant professors generally having the largest averages.

Not surprisingly, the majority of the departments have increased the number of female faculty by hiring assistant professors. The question of how the pool of female Ph.D.'s has changed is addressed in the next section.

Trends in Ph.D. Production

The number of Ph.D.'s awarded in the U.S. has seen a significant increase since 2002. According to the Taulbee Survey, U.S. Ph.D. production doubled from 2002 to 2008, from about 700 to 1,400. The proportion of females receiving a Ph.D. has increased. For all 163 departments, the increase in proportion went from 15.28% in 1999 to 19.96% in 2009. Due to the increase in overall number and proportion, the number of females receiving a Ph.D. has more than doubled since 2002. Figure 3 shows the proportion and the difference from the baseline average for each of the eight groups.

The proportion of Ph.D.'s awarded to females exhibits some similarities with the trends seen for female faculty. In particular, the figure shows an upward trend for all groups. Although there are differences between the groups, there does not appear to be a group that does significantly better (or worse) than the average. Not surprisingly, the lines representing the proportions are more irregular as the number of Ph.D.'s generated each year generally fluctuates. However, the proportions in the two panel rows in Figure 3 appear to exhibit higher proportions for the groups in 37+. Indeed this is the case. For example, in 2009, the representation of females receiving a Ph.D. is 22% for departments in group 37+ and 18% for departments in group 1–36.

The proportion of Ph.D.'s awarded to females exhibits some similarities with the trends seen for female faculty.

Figure 4 shows the total number of Ph.D.'s awarded (to male and female students) in departments in group 1–36 (orange solid) and the total number awarded in departments in group 37+ (orange dashed). The departments in groups 1–36 and 37+ produce about the same number of Ph.D's. In 2009, the 127 departments in group 37+ produced a total of 687 Ph.D.'s and 36 departments in group 1–36 produced a total of 650 Ph.D.'s. As expected, the average number of Ph.D.'s produced by a department in group 1–36 is significantly higher; it is 21 Ph.D.'s per year versus eight for departments in group 37+. However, the set of departments in group 37+ awards more Ph.D.'s to females in total numbers as well as proportions.

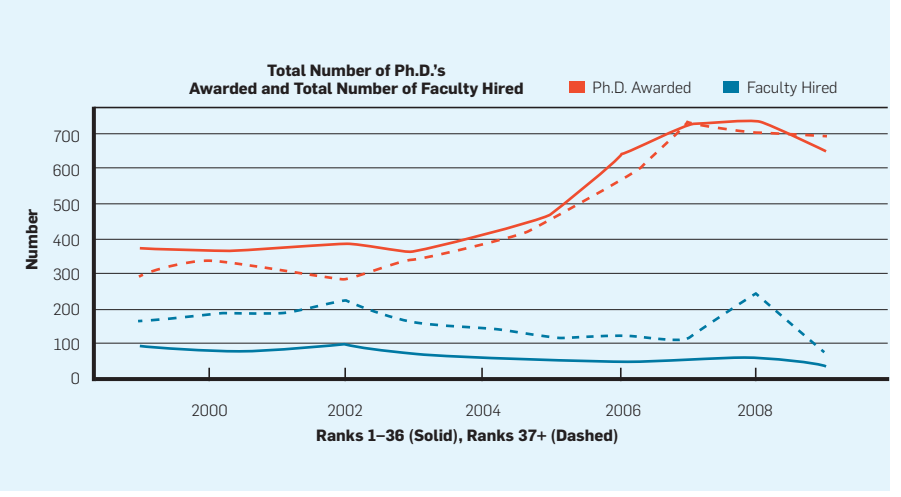
Figure 4 also shows total Ph.D. production versus faculty hiring. The blue lines show faculty hired by departments in the groups 1–36 (solid) and 37+ (dashed). The data supports what many departments have experienced: since 2002, the number of new Ph.D.'s seeking an academic position has dramatically increased while the number of tenure-track faculty hired has decreased. The peak of the dashed purple line in 2008 represents a hiring spurt in group 37+; the full report shows that it happened in Public 37+ Large.

Trends in Bachelor's Degree Production

Ph.D.-granting institutions in the U.S. generate only a fraction (less than 20%) of the undergraduate degrees in computer science; however, trends found in the data for Ph.D.-granting institutions are typically also present in data for all bachelor's degrees. Such data is available from WebCASPAR (https://webcaspar.nsf.gov/). Since 2003, there has been a drop in total bachelor's degree production, both for both male and female students. In addition, there has been a decline in the proportion of females receiving B.S. degrees: from 18% in 2000 to 10% in 2008.

Figure 5 shows the total numbers for undergraduate degree production for males and female students for public versus private institutions. The data suggests a peak in the number of bachelor's degrees awarded between 2003 (for females) and 2004 (for males). The total number of degrees awarded in the public schools is considerably larger than that in the private schools, but the average counts

Figure 4. Total number of Ph.D.'s awarded and total number of faculty hired for all departments in group ranks 1–36 (solid) and ranks 37+ (dashed).



are not so disparate (see the full report for supporting plots).

Figure 6 shows the proportions of female students receiving a bachelor's degree in private and public institutions, with the left panel showing the proportions for all institutions ranked 1–36 and the right panel for institutions 37+. The declining trend in proportions starts at different times, with public institutions experiencing it earlier than the private ones. Since 2004, the private institutions in group 1–36 have an increasingly higher percentage of female undergraduates than public ones. The relatively low response rate to the Taulbee Survey from departments in group Private 37+ most likely impacts the irregular shape of the corresponding line. The full report indicates B.S. enrollment in Ph.D.-granting departments appears to have stabilized, but no significant overall increases can be observed.

Conclusion

We have examined the last 10 years of Taulbee Survey data to investigate claims about gender representation in U.S. computer science departments, at all stages of the pipeline. The data shows increases in both female faculty (tenured and tenure-track) and female Ph.D. graduates, but a decline in females receiving B.S. degrees. In aggregate, these findings are consistent with current views of the state of CS. However, surprisingly, our analysis also shows the changes are not uniform with respect to department type. This leads to a number of interesting questions.

First, there are two questions related to the representation of females:

Figure 5. Total number of males (dashed) and females (solid) awarded bachelor's degrees in public and private schools.

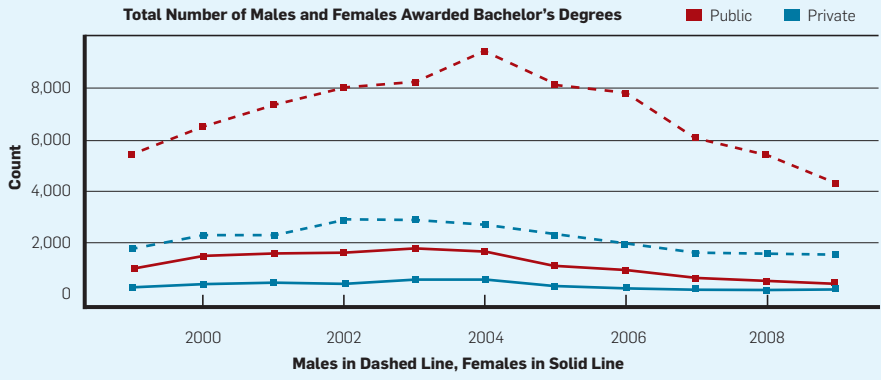
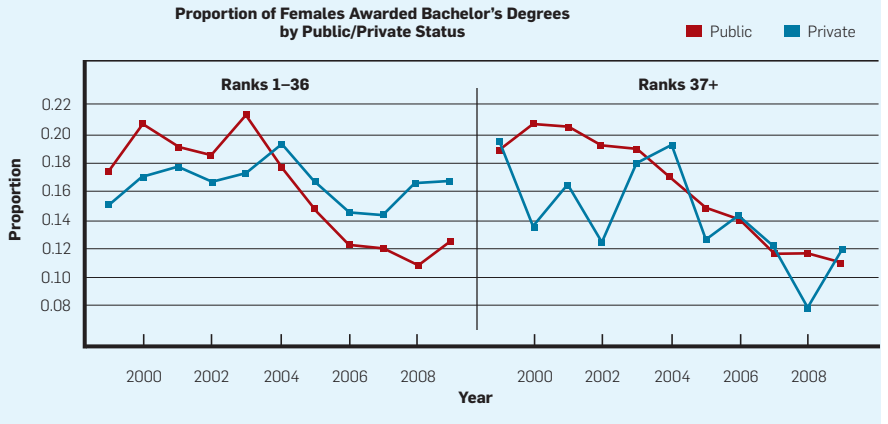


Figure 6. Proportion of female students awarded bachelor's degrees in private versus public institutions.



Future studies of trends should differentiate Ph.D.-granting departments on more dimensions than just rank in order to gain a better understanding of gender demographics.

Why do many top-ranked departments have so few female assistant professors? Why do lower-ranked departments graduate a higher percentage of female Ph.D.'s? Second, the analysis highlights the groups that have successfully improved gender representation: What strategies did departments in Public 1–36 2nd have to successfully hire female faculty? Did they attract more female applicants or did they approach the process of hiring differently? What strategies did Private 1–36 departments use to increase the proportion of female undergraduates? Did they attract new students to CS or did they attract existing CS students to their institution?

Our overall analysis illustrates that future studies of trends should differentiate Ph.D.-granting departments on more dimensions than just rank in order to gain a better understanding of gender demographics.

Douglas Baumann (dbaumann@purdue.edu) is a research assistant in the Department of Statistics at Purdue University, West Lafayette, IN.

Susanne Hambrusch (seh@purdue.edu) is a professor of computer science at Purdue University, West Lafayette, IN.

Jennifer Neville (neville@purdue.edu) is an assistant professor of computer science and statistics at Purdue University, West Lafayette, IN.

We thank Betsy Bizot from CRA for generating the underlying data sets. Without her support and guidance in understanding the data, the report on which this Viewpoint is based could not have been developed.

PLEASE NOTE CHANGE OF DATE

DIS

DESIGNING INTERACTIVE SYSTEMS 2012

Newcastle, UK
June 11–15, 2012

Call for Contributions

Designing Interactive Systems 2012 addresses design as an integrated activity, showcasing research that explores the technical, social, cognitive, organizational, and cultural factors of design.

DIS 2012 turns its focus to 'In the Wild', promoting exchange and discussion on the opportunities, challenges and issues of interactive systems in the everyday practice and lived experience of people and institutions.

Over five days attendees will be invited to share research, innovation, best practices and learning through a range of avenues including workshops, demonstrations, invited talks, and a new addition this year, design lunch dates that aim to promote networking among newer and more experienced members of the interaction design community.

Submission Deadlines

Workshop proposal deadline	December 09, 2011
Full and short paper submission deadline	January 20, 2012
Demonstration and doctoral consortium deadline	March 07, 2012

PERVASIVE 2012

Newcastle will also host Pervasive 2012 on June 18–22

See www.pervasiveconference.org for details

www.dis2012.org

Culture Lab Newcastle

Newcastle University

acm Association for Computing Machinery

DOI:10.1145/2018396.2018412

Article development led by acmqueue
queue.acm.org

The time has come for software liability laws.

BY POUL-HENNING KAMP

The Software Industry *is* the Problem

ILLUSTRATION BY ALEX WILLIAMSON

ONE SCORE AND seven years ago, Ken Thompson brought forth a new problem, conceived by thinking, and dedicated to the proposition that those who trusted computers were in deep trouble.

I am, of course, talking about Thompson's 1984 ACM A.M. Turing Award Lecture—"Reflections on Trusting Trust."[2] Unless you remember this piece by heart, you might want to take a moment to read it if at all possible (http://bit.ly/nNGh5b).

The one sentence in Thompson's lecture that really, *really* matters is: "You can't trust code that you did not totally create yourself."

This statement is not a matter of politics, opinion, taste, or in any other way a value judgment; it is a fundamental law of nature, which follows directly from pure mathematics in the general vicinity of the works of Turing and Gödel. If you doubt this, please (at your convenience) read Douglas Hofstadter's classic *Gödel, Escher, Bach*,[1] and when you get to the part about "Mr. Crab's *record player*," substitute "Mr. Crab's *laptop*."

Gödel, Escher, Bach

Hofstadter's book, originally published in 1979, does not in any way detract from Ken Thompson's fame, if, indeed, his lecture was inspired by it;

1979 was a long time ago, and it's possible that not every reader may know of—much less have read—this book. My editor proposed that I summarize or quote from it to make things clearer for such readers.

Considering that Gödel, Escher, and Bach are all known for their intricate multilayered works and that Hofstadter's book is a well-mixed stew not only of their works, but also of the works of Cantor, Church, Gantõr, Turing, and pretty much any other mathematician or philosopher you care to mention, I will not attempt a summary beyond: "It's a book about how we think."

The relevant aspect of the book here is Gödel's incompleteness theorem, which, broadly speaking, says that no finite mathematical system can resolve, definitively, the truth value of all possible mathematical conjectures expressible in that same

mathematical system.

In the book this is illustrated with a fable about Mr. Crab's "perfect record player," which, because it can play any and all sounds, can also play sounds that make it resonate and self-destroy—a vulnerability exploited on the carefully constructed records of Mr. Crab's adversary, Mr. Tortoise.

Mr. Crab tries to protect against this attack by preanalyzing records and rearranging the record player to avoid any vulnerable resonance frequencies, but Mr. Tortoise just crafts the sounds on his records to the resonance frequencies of the part of the record player responsible for the rearrangement. This leaves Mr. Crab no alternative but to restrict his record playing to only his own, preapproved records, thereby severely limiting the utility of his record player.

Malware-scanning programs try to classify executable code into "safe" and "unsafe," instead of mathematical conjectures into "true" and "false," but the situation and result are the same: there invariably is a third pile called "cannot decide either way," and whatever ends up in that pile is either a security or a productivity risk for the computer user.

Amusingly, malware scanners almost unfailingly classify malware-scanner programs, including themselves, as malware, and therefore contain explicit exemptions to suppress these "false" positives. These exemptions are of course exploitable by malware—which means the classification of malware scanners as malware was correct to begin with. "Quis custodiet ipsos custodes?" (Who will guard the guards themselves?)

Back to Thompson

In 1984, the Thompson lecture evoked wry grins and minor sweating for Unix system administrators at universities, because those were the only places where computers were exposed to hostile users who were allowed to compile their own programs. Apart from sporadic and mostly humorous implementations, however, no apocalyptic horsemen materialized in the sky.

In recent years, there have been a number of documented instances where open source projects were broken into and their source code modified to add backdoors. As far as I am aware, none of these attacks so far has reached further than the lowest rung on Ken Thompson's attack ladder in the form of a hardcoded backdoor, clearly visible in the source code. Considering the value to criminals, however, it is only a matter of time before more advanced attacks, along the line Thompson laid out, will be attempted.

In strict mathematical terms, you cannot trust a house you did not totally create yourself, but in reality, most of us will trust a house built by a suitably skilled professional.

The security situation with commercial closed-source software is anyone's guess, but there is no reason to think—and no credible factual basis for a claim—that the situation is any different or any better than it is for open source projects.

The now-legendary Stuxnet malware incident has seriously raised the bar for just how sophisticated attacks can be. The idea that a widely deployed implementation of Java is compiled with a compromised compiler is perfectly reasonable. Outsourced software development does not make that scenario any less realistic, likely, or scary.

We Have to Do Something, But What?

We have to do something that actually works, as opposed to accepting a security circus in the form of virus or malware scanners and other mathematically proven insufficient and inefficient efforts. We are approaching the point where people and organizations are falling back to pen and paper for keeping important secrets, because they no longer trust their computers to keep them safe.

Ken Thompson's statement—"You can't trust code that you did not totally create yourself"—points out a harsh and inescapable reality. Just as we don't expect people to build their own cars, mobile phones, or homes, we cannot expect secretaries to create their own text-processing programs nor accountants to create their own accounting systems and spreadsheet software. In strict mathematical terms, you cannot trust a house you did not totally create yourself, but in reality, most of us will trust a house built by a suitably skilled professional. Usually we trust it more than the one we might have built ourselves—even when we may have never met the builder and/or when the builder is

dead. The reason for this trust is that shoddy construction has had negative consequences for builders for more than 3,700 years. "If a builder builds a house for someone, and does not construct it properly, and the house which he built falls in and kills its owner, then the builder shall be put to death." (Hammurabi's Code, approx. 1700 BC)

Today the operant legal concept is "product liability," and the fundamental formula is "if you make money selling something, you'd better do it properly, or you will be held responsible for the trouble it causes." I want to point out, however, that there are implementations of product liability other than those in force in the U.S. For example, if you burn yourself on hot coffee in Denmark, you burn yourself on hot coffee. You do not become a millionaire or necessitate signs pointing out that the coffee is hot.

Some say the only two products not covered by product liability today are religion and software. For software that has to end; otherwise, we will never get a handle on the security madness unfolding before our eyes almost daily in increasingly dramatic headlines. The question is how to *introduce* product liability, because just imposing it would instantly shut down any and all software houses with just a hint of a risk management function on their organizational charts.

A Software Liability Law

My straw-man proposal for a software liability law has three clauses:

Clause 0. Consult criminal code to see if any intentionally caused damage is already covered. I am trying to impose a civil liability only for unintentionally caused damage, whether a result of sloppy coding, insufficient testing, cost cutting, incomplete documentation, or just plain incompetence. Intentionally inflicted damage is a criminal matter, and most countries already have laws on the books for this.

Clause 1. If you deliver software with complete and buildable source code and a license that allows disabling any functionality or code by the licensee, then your liability is limited to a refund. This clause addresses how to avoid liability: license your users to inspect and chop off any and all bits of your software they do not trust or do not want to run, and make it practical for them to do so.

The word *disabling* is chosen very carefully. This clause grants no permission to change or modify how the program works, only to disable the parts of it that the licensee does not want. There is also no requirement that the licensee actually look at the source code, only that it was received.

All other copyrights are still yours to control, and your license can contain any language and restriction you care to include, leaving the situation unchanged with respect to hardware locking, confidentiality, secrets, software piracy, magic numbers, and so on. Free and open source software is obviously covered by this clause, and it does not change its legal situation in any way.

Clause 2. In any other case, you are liable for whatever damage your software causes when used normally. If you do not want to accept the information sharing in Clause 1, you would fall under Clause 2 and have to live with normal product liability, just as manufacturers of cars, blenders, chainsaws, and hot coffee do. How dire the consequences and what constitutes "used normally" are for the legislature and courts to decide.

An example: A salesperson from one of your longtime vendors visits and delivers new product documentation on a USB key. You plug the USB key into your computer and copy the files onto the computer. This is "used normally" and should never cause your computer to become part of a botnet, transmit your credit card number to Elbonia, or send all your design documents to the vendor.

The majority of today's commercial software would fall under Clause 2. To give software houses a reasonable chance to clean up their acts and/or to fall under Clause 1, a sunrise period would make sense, but it should be no longer than five years, as the laws would be aimed at solving a serious computer security problem.

And that is it, really. Software houses will deliver quality and back it up with product liability guarantees, or their customers will endeavor to protect themselves.

Would it Work ?

There is little doubt that my proposal would increase software quality and computer security in the long run, which is exactly what the current situation calls for.

It is also pretty certain there will be some short-term nasty surprises when badly written source code gets a wider audience. When that happens, it is important to remember that today the good guys have neither the technical nor legal ability to know if they should even be worried, as the only people with source-code access are the software houses and the criminals.

The software houses would yell bloody murder if any legislator were to introduce a bill proposing these stipulations, and any pundit and lobbyist they could afford would spew their dire predictions that "this law will mean the end of computing as we all know it!"

To which my considered answer would be: "Yes, please! That was exactly the idea." C

Related articles on queue.acm.org

CTO Roundtable: Malware Defense
http://queue.acm.org/detail.cfm?id=1731902

All Things Being Equal?
Stan Kelly-Bootle
http://queue.acm.org/detail.cfm?id=1348596

B.Y.O.C. (1,342 Times and Counting)
Poul-Henning Kamp
http://queue.acm.org/detail.cfm?id=1944489

References

1. Hofstadter, D. *Gödel, Escher, Bach.* Basic Books, 1999.
2. Thompson, K. Reflections on trusting trust. *Commun. ACM 27*, 8 (Aug. 1984), 761–763; http://m.cacm.acm.org/magazines/1984/8/10471-reflections-on-trusting-trust/pdf.

Poul-Henning Kamp (phk@FreeBSD.org) has programmed computers for 26 years and is the inspiration behind bikeshed.org. His software has been widely adopted as "under the hood" building blocks in both open source and commercial products. His most recent project is the Varnish HTTP accelerator, which is used to speed up large Web sites such as Facebook.

DOI:10.1145/2018396.2018411

Article development led by acmqueue
queue.acm.org

Difficult technical problems and tough business challenges.

BY LI GONG

Java Security Architecture Revisited

THE JAVA PLATFORM JDK 1.0 was released in 1995 with a simplistic all-or-nothing "sandbox" security model. Li Gong joined the JavaSoft division of Sun Microsystems in 1996 and led the redesign of the security architecture that was first released in 1998 as JDK 1.2 and is now deployed on numerous systems and devices from the desktop to enterprise and mobile versions of Java.

This article looks back at a few of the most difficult technical problems from a design and engineering perspective, as well as some tough business challenges for which research scientists are rarely trained. Li offers a retrospective here culled from four previous occasions when he had the opportunity to dig into old notes and refresh his memory: 2002 Workshop on the Economics of Information Security, Berkeley, CA; 2003 UW/MSR/CMU Summer Institute, Stevenson, Washington; ACM's 2009 Computer Security Applications Conference, Honolulu; and a seminar last May at the University of Cambridge Computer Laboratory, England.

Although security architects are not "in business," it is important that they are clear about who their customers are. They rarely build directly for individual end users, who do not directly use the operating system, although they are often the eventual beneficiaries.

Most of the work of a security architect is targeted at application programmers, and Java is no exception. Here the design goal is to help programmers get what is intended out of their code—more specifically, to make the most common cases the easiest to write and get right, and to reduce the risk of coding mistakes or bugs. As such, the four attributes of the Java security architecture[1] should generally apply:

- **Usability.** To be ubiquitous and accepted in the marketplace, the architecture must be easy to use and suitable for writing a wide variety of applications.
- **Simplicity.** To inspire confidence in the correctness of the architecture, it must be easy to understand the critical design properties and to analyze the implementation.
- **Adequacy.** The architecture must contain all essential features and building blocks for supporting higher-level security requirements.
- **Adaptability.** The design must evolve with ease, following demand and market reality. In particular, it should avoid overprescribing that restricts programmability.

In hindsight, having these guiding principles in place was crucial. In the original JDK 1.0, the security mechanism was all about special casing—code being inside versus outside the sandbox. That seemingly simple architecture paradoxically resulted in complicated design, fragile code, and numerous security bugs. In JDK 1.2, security was designed to be general, systematic, and simple-minded, and this resulted in a more robust and usable platform. We fought off a competing design from Netscape that was specialized for browser usage. Our design is not only broad in scope, covering desktops, servers, and embedded and mobile de-

vices, but also specific enough to enable programmers to build browser-centric applications. I will return to these topics later.

Guess Who's Coming to Dinner

The design aspiration is to ensure that Java code is executed as intended without undesirable side effects. This goal has three components. The first is to ensure that only valid Java code is accepted; this is the topic of the current section. The second is to ensure that intended behavior occurs as designed; this is usually taken care of via testing and is well understood, and therefore is not dealt with further here. The third is to prevent bad unintended behavior, such as access to critical data that should not have been allowed; this is dealt with later in the section on the principle of least privilege.

Yet another often-implicit requirement is that all the checks and balances must be done reasonably fast—meaning the system has performance characteristics comparable to that of a system with no security mechanism at all. The threat model here is focused primarily on untrusted code that might engage in malicious actions. The protection mechanism aims to stop those malicious behaviors; it also helps reduce risks of benign coding mistakes, although it cannot expect to protect against all faulty programming practices, such as not validating queries that might lead to SQL injection attacks.

Typically, an application is written in Java source code, which is compiled into platform-independent Java bytecode, which is then executed by the JVM (Java Virtual Machine). It is possible to compile Java source code directly into machine-specific native code (such as for x86 systems). This scenario is not discussed further here because compiled native code bypasses the Java mechanism and cannot be dealt with entirely within the Java platform except by way of allowing or disallowing native code access and execution. It is also possible to write Java bytecode directly, although most people choose not to practice this special art. In any event, even bytecode generated by compilers may not be trusted. In fact, Ken Thompson went much further in his well-known 1984 Turing Award lecture, "Reflections on Trusting Trust," by saying, "You can't trust code that you did not totally create yourself." Thus, the JVM must be able to decide if a piece of bytecode is valid and acceptable.

Each unit of Java bytecode (the opcode) is exactly one byte long and is well defined. A truthful compiler takes valid Java source code and produces a sequence of bytecode that accurately re-

ILLUSTRATION BY MATTHEW COOPER

flects the intentions of the source code and at the same time maintains the inherent properties of the Java language, such as type safety. A maliciously generated sequence of bytecode, on the other hand, may not correspond to any valid Java source code at all and can intentionally break language properties in order to enable security attacks.

Telling whether a presented series opcode is "valid" is fundamentally a form of the input validation problem. Suppose the JVM takes any integer in the range of 1 to 9 as valid input; then input validation is trivial. In reality, the input space that contains arbitrary sequences of opcode is unlimited in size and sparsely populated with valid bytecode sequences. Because there is no simple validation test formula through which to run a target code, the Java runtime system does bytecode validation in multiple stages, with various techniques, at different places. The bytecode verifier statically checks incoming code. Once the code is inside the system, type-safety mechanisms are deployed throughout the JVM to spot and stop illegitimate code. All these maneuvers are complex, and, in the absence of formal verification of the total system, there is no way to know for sure that all possible invalid codes can be spotted.

Thus, one really hard problem for any runtime system that deals with executable code compiled from high-level languages is to ensure that code received from "foreign" sources is valid input. For most programming languages, this is simply not possible. Java's platform-independent bytecode makes this task possible, but it is still extremely difficult to get right. Brian Bershad and his (then) student Emin Gun Sirer at the University of Washington came up with the concept of a bytecode basher.[4] They set up an automated system to generate random sequences of opcode to throw at any particular Java runtime system; they watch to see if the system breaks and then analyze the results to figure out the flaws in the Java implementation. This randomized and automated approach is a surprisingly low-cost yet effective tool that the JavaSoft team quickly adopted.

Who Moved My Cheese?

Now comes the problem of preventing bad unintended behavior. For example, when a Java application or applet triggers an access request for a local file, should the request be granted? Well, it depends. If the request is to read the local file containing personal credit card information, then the request may or may not be granted, depending on if there is a security policy or user preference in place. If the request is to read the font file for 12-point Calibri type so that a text file can be displayed according to the word processor specification, then it almost always should be granted, because, implicitly, font files are structured to be harmless if used in this fashion. Note that applications never directly open files. They call the file APIs in the Java platform for these operations. These APIs have no built-in notion of security, except that they (or their designers) know that file operations are sensitive, so they better make a call to the SecurityManager for consultation.

Here the problem rears its ugly head—the SecurityManager is put on the spot and has no clothes on. For example, it is quite alright for the display code written in the system to access the font file, but usually it is a bad idea for the application to gain direct access to system font files, because these files could be arbitrarily changed and that might lead to future display problems. The SecurityManager can hardly differentiate between these two scenarios, however, let alone the indefinite number of variations.

In JDK 1.0/1.1, the all-or-nothing sandbox security model works more or less as follows. The SecurityManager looks up the call history (of method invocations). If all code is local (that is, no remote-loaded and therefore untrusted applets), then the access request is granted. If an applet is present in the call chain, then the request is denied, unless the immediate caller to access the file is system code; *except* when that code should not be accessing font files but not when the applet code is in fact in a call-back situation; except when the system code that makes the call back (to the applet) should not really access font files, and so on and so on. Moreover, what about threads, exceptions, and other constructs that mix up or disconnect the execution context? You get the picture.

Fundamentally, trying to guess a program's intention is impractical. You would be much better off requiring programmers to declare their intentions explicitly, as we shall see later.

To make matters worse, the SecurityManager implementation does not actually run through this logical deduction—it cannot. Instead, based on some rules of thumb and a particular instance of the Java system, the SecurityManager simply counts the *distance*—the number of method calls—between itself and the nearest applet code, and heuristically makes a decision on whether a request should be granted. It should be obvious now that this setup means that whenever a part of the system is changed, the heuristics can become wrong and thus must be adjusted, regardless of whether the heuristics were correct or complete in the first place, and regardless of the method used to extend the system to include a new concept such as users/principals in making access-request decisions. This fragile setup was the source of many security holes.

Security in JDK 1.2 was rearchitected completely, adopting the well-known but almost never practiced principle of least privilege.[3] All code is treated equally. Each piece of code is given a set of privileges (access rights), either explicitly (through policy, administration, or preferences) or implicitly (where system code has full privileges, while applets have the same level of privilege as in their sandbox days). At any point in execution, an access request is granted if each piece of code along the call chain has sufficient privilege for the requested access. In other words, the effective set of privileges is the intersection of all privilege sets for the code along the call history—the principle of least privilege. Moreover, context information pertaining to security can be encapsulated and passed along so that one cannot fool the system by spawning new threads or throwing exceptions. All of these assume, of course, that the code that implements the security mechanism is itself secure and correct.

A piece of code—say, the display library that may need access to font files from time to time—can explicitly declare to exercise its own privilege unilaterally, telling the security system to ignore codes that have come before it. This design lets programmers, especially those who write system and library

code, to explicitly declare their intentions when performing sensitive operations. This is akin to the setuid feature in Unix except that, instead of enabling system-high privilege for the entire program, in Java the privileged mode lasts only as long as the duration of the privileged method call. Note that if this privileged code later calls less privileged code, the effective set of privileges will still be curtailed because of the principle of least privilege.

The need for explicit declaration may appear cumbersome at first, but programmers can rest assured that their code will not unintentionally weaken security. Furthermore, the majority of programs do not need to invoke their privileges specifically, so we have given the most common programming cases the best of both worlds—ease of coding and peace of mind. Note that it may not be easy for programmers to figure out exactly which of their privileges they need to declare in order to make their programs work properly in all possible scenarios. The JDK 1.2 design actually does not support fine-grained privilege declarations. A declaration enables all the privileges associated with the code.

The major lesson here is that being systematic is easier and more robust than being ad hoc, though not everyone understands this. Toward the end of JDK 1.2 development, during a security audit of the entire code base (which got enacted after much begging, plus sticks and carrots), we discovered that a Sun engineer working on JDK had deliberately duplicated and then modified system code such that his own library code would not have to bother with an explicit declaration of privileges—a move that may have made his job slightly easier but would have led to serious security breaches for all users of the Java platform if his misdeed had gone undetected.

A number of hard problems remain in this area. Top among them is whether least privilege calculations can be done efficiently, especially with very complex security parameters—for example, complicated access-control policies, many different types of access rights, and an intricate execution environment. Another problem is that assigning different privileges to different code created complexities for other parts of the system. For example, optimizations done by JIT (just-in-time) compilers must now conform to additional security requirements.

A piece of code can explicitly declare to exercise its own privilege unilaterally, telling the security system to ignore codes that have come before it. This design lets programmers to explicitly declare their intentions when performing sensitive operations.

Yet another perennial problem is the practical side of security policy management and deployment. A more theoretical question, but one nonetheless worth pondering, is the scope of security policies that can (or cannot) be enforced with the least-privileged model, using the rather conventional categories of access-control permission types defined in JDK 1.2. Fred Schneider of Cornell University developed an intriguing concept called Inline Reference Monitor and proved that it can express and enforce any and all policies enforceable by monitoring system execution.[5]

Despite these difficult issues, one comforting thought may be that, after more than 12 years in the field, the principle of least privilege as architected in JDK 1.2 has stood the test of time and probably saved untold numbers of coding mistakes from turning into security blunders.

The Importance of Being Earnest

Many other technical lessons are worth repeating periodically. For example, you should be very judicious about the use of NULL, because you cannot change the behavior of nothing. In JDK 1.0/1.1, in certain circumstances, the `ClassLoader` or `SecurityManager` could be `NULL`, which made it difficult to retrofit a more fine-grained design.

As another example, during runtime Java turns static code into live objects. This process actually contains two separate steps: locate the code description; define it into a live object. The first step should be open and extensible in nature, because both the runtime system and applications should be able to specify desired locations for obtaining code. The second step, on the other hand, must be strictly controlled so that only trusted system code can handle the job of creating objects. Unfortunately, these two steps were overloaded into a single method, which worked well in the all-or-nothing model but caused much difficulty when JDK 1.2 changed into a more nuanced world.

Another issue may be surprising to many people: strictly speaking, Java cannot guarantee sequential execution of consecutive instructions. One

simple reason is that exceptions can be thrown, causing the execution thread to detour (and may never return). One remedy is to use clauses such as `Try/Finally` to force a return. In more extreme cases—for example, when the actual physical machine runs out of memory—the behavior of the Java runtime system is undefined and certainly nowhere near being failsafe. These situations are further complicated by the fact that many key JVM functionalities, including some for security, are written in Java, so problems in one part of the system could easily impact the correctness of another part of the system. For all these design challenges and alternatives, please refer to the Java security book[2] and latest Java documentation.

The remainder of this article addresses the challenges that were entirely unexpected for someone whose previous work experience was confined to the world of academia. Scientists and engineers are trained to tackle technical problems, but real-world projects—especially one with industry-wide impact such as Java—are equally social and political. In roughly 30 months of working on JDK, I attended around 1,000 meetings and took 300-plus pages of notes. One can easily forget the war-zone atmosphere back then, especially the Friday fire drills. Too often, (outside) security researchers would inform us of newly discovered security holes on Friday and give us until Monday at noon to figure out a patch and response, when they would inform *The New York Times*, *Wall Street Journal*, and other media. Sometimes leaks to journalists occurred right after we rolled out patches to Java licensees (including IBM, Microsoft, Netscape, and many others), and we could only guess which of them had the motivation to publicize the security holes before patches were put in place.

Then there was a whole assortment of other equally time- and energy-consuming distractions, such as U.S. export control regulations on basic cryptography (since relaxed), patents on RSA and public-key technologies (since expired), and issues such as code obfuscation, Java for e-commerce and smart cards, and JavaOS.

To make sure we were on the right path, we invited a small number of academic and industry experts (including Jerome Saltzer of MIT and Michael Schroeder from DEC Systems Research Center, authors of the original principle of least privilege paper) and convened a formal Java Security Advisory Council, which provided regular reviews and valuable feedback as the rearchitecting progressed. We also received great advice from many sources, mainly academic researchers and industry partners—not all of which was solicited or friendly. A few strong-headed researchers wanted their alternative designs incorporated into the Java platform and threw various threats at us.

Netscape was a unique story. It was the most popular browser to include Java and therefore was a valued partner; it also had its own ideas about where Java should be headed, and those ambitions made the relationship difficult. On a technical level, the main dispute in the area of security was between Netscape's notion that Java was basically just a browser component so security mechanisms should be geared toward browser users, and our vision that Java was a general programming platform that should cater to all kinds of uses, including browsers and server-side applications.

On an engineering level, Netscape was innovating and shipping a new version every three months, while Sun/JavaSoft would take a year or two to ship a major release with new features (such as those requested by Netscape) that would become available through the official JDK platform. As the divergence between Java code in JDK and that in the Netscape browser was becoming unmanageably large, the presidents of Netscape and JavaSoft invited IBM to perform a confidential and binding arbitration. After months of intensive fact finding, code collecting, and *Consumer Reports*-style scoring, IBM called a resolution meeting at IBM's Java building, a block away from JavaSoft, on Oct. 15, 1997, and announced that JavaSoft's design had won.

Looking back after so many years, I can see at least three lasting effects of the Java security work. The most obvious is that the new security architecture provided better support for Java programmers to make their applications more secure and to reduce risks when the code was faulty. Second, we raised the bar for everyone else in the sense that any new language or platform must consider type safety, systems security, and the principle of least privilege, because we have demonstrated that these are achievable even in a large-scale commercial setting. Finally, the security constructs in Java have increased security awareness for thousands of developers who can then transfer this knowledge to other programming languages and development platforms.

Acknowledgments

I'd like to thank Jeannette Wing of Carnegie Mellon University, Jeremy Epstein and Peter Neumann of SRI International, and Ross Anderson and Robert Watson at the University of Cambridge for inviting me to give those retrospective talks on Java security. I am grateful to Robert Watson and Jim Maurer at ACM for encouraging me to write up the Cambridge talk for *Communications*, and to the thoughtful anonymous reviewers. I am, of course, deeply in debt to all the people who have cared for, helped with, and supported the Java security project.

Related articles on queue.acm.org

An Open Web Services Architecture
Stan Kleijnen, Srikanth Raju
http://queue.acm.org/detail.cfm?id=637961

How OSGi Changed My Life
Peter Kriens
http://queue.acm.org/detail.cfm?id=1348594

Untangling Enterprise Java
Chris Richardson
http://queue.acm.org/detail.cfm?id=1142045

References

1. Gong, L. Java security: present and near future. *IEEE Micro* (May 1997), 14–19.
2. Gong, L., Ellison, G. and Dageforde, M. *Inside Java 2 Platform Security: Architecture, API Design and Implementation*, second ed. Addison-Wesley, Reading, PA, 2003.
3. Saltzer, J. H. and Schroeder, M.D. The protection of information in computer systems. *Commun. ACM 17*, 7 (July 1974).
4. Sirer, E. and Bershad, B. Testing Java Virtual Machines. In *Proceedings of the International Conference on Software Testing and Review* (Nov. 1999).
5. Schneider, F.B. Enforceable security policies. *ACM Trans. Information and System Security* (Feb. 2000), 30–50.

Li Gong is chairman and CEO of Mozilla Online Ltd., the Beijing-based Mozilla subsidiary. He was formerly a Distinguished Engineer and the chief Java security architect at the JavaSoft division of Sun Microsystems. Two of his patents on Java security are among the seven that were the focus of a lawsuit between Oracle and Google over Android in 2010.

DOI:10.1145/2018396.2018413

Article development led by acmqueue
queue.acm.org

Why the next language you learn should be functional.

BY YARON MINSKY

OCaml for the Masses

SOMETIMES, THE ELEGANT *implementation is a function. Not a method. Not a class. Not a framework. Just a function.* —*John Carmack*

Functional programming is an old idea with a distinguished history. Lisp, a functional language inspired by Alonzo Church's lambda calculus, was one of the first programming languages developed at the dawn of the computing age. Statically typed functional languages such as OCaml and Haskell are newer, but their roots go deep—ML, from which they descend, dates back to work by Robin Milner in the early 1970s relating to the pioneering Logic for Computable Functions (LCF) theorem prover.

Functional programming has also been enormously influential. Many fundamental advances in programming language design, from garbage collection to generics to type inference, came out of the functional world and were commonplace there decades before they made it to other languages.

Yet functional languages never really made it to the mainstream. They came closest, perhaps, in the days of Symbolics and the Lisp machines, but those days seem quite remote now. Despite a resurgence of functional programming in the past few years, it remains a technology more talked about than used.

It is tempting to conclude from this record that functional languages do not have what it takes. They may make sense for certain limited applications, and contain useful concepts to be imported into other languages; but imperative and object-oriented languages are simply better suited to the vast majority of software engineering tasks.

Tempting as it is, this conclusion is wrong. I have been using OCaml in a production environment for nearly a decade, and over that time I have become convinced that functional languages, and in particular, statically typed ones such as OCaml and Haskell, are excel-

lent general-purpose programming tools—better than any existing mainstream language. They also have an enormous range, being well suited for small scripting tasks, as well as large-scale high-performance applications. They are not the right tool for every job, but they come surprisingly close.

The Move to OCaml

Most of my experience programming in OCaml came through my work at Jane Street, a financial firm founded in 2000. Nine years ago, no one at Jane Street had heard of OCaml. Today, Jane Street is the biggest industrial user of the language, with nearly two million lines of OCaml code and, at last count, 65 employees who use the language on a daily basis. Probably the best way to explain what makes OCaml such an effective tool is to start by explaining how and why that transformation took place. To understand that, you first need to understand something about what Jane Street does.

Jane Street's core business is providing liquidity on the world's electronic markets. It is, essentially, a middleman. It continually places orders for many different securities on many different exchanges. Each order expresses a willingness to either buy or to sell a given security at a given price, and, collectively, they are an advertisement to the markets of Jane Street's services. Through these orders, the firm buys from people who need to sell and sells to people who need to buy, making money from the gap between the buying and selling prices. All the time it is competing on price with other players trying to do the same thing.

Electronic liquidity provision is technologically intense, not only because of the computational resources that need to be deployed (an enormous amount of data needs to be consumed, analyzed, and responded to in real time), but also in terms of the complexity of the enterprise—trading can cross multiple exchanges, regulatory regimes, security classes, and time zones. Managing that complexity is a daunting task that requires a significant investment in software.

All this technology carries risk. There is no faster way for a trading firm to destroy itself than to deploy a piece of trading software that makes a bad decision over and over in a tight loop. Part of Jane Street's reaction to these technological risks was to put a very strong focus on building software that was easily understood—software that was readable.

Programmers who are new to OCaml are often taken aback by the degree to which the type system catches bugs.

Reading code was part of the firm's approach to risk from before we had written our first line of OCaml. Early on, a couple of the most senior traders (including one of the founders) committed to reading every line of code that went into the core trading systems before those systems went into production. This was an enormous ongoing time investment and reflected the high level of concern about technology risk.

I started at Jane Street the year after I finished my Ph.D., working there part-time while doing a post-doc. My work there was focused on statistical analysis and optimization of trading strategies, and OCaml was the primary tool I used to get the analysis done. Why OCaml? I had learned it in grad school and fell in love with the language then. And OCaml was a great match for this kind of rapid-prototyping work: highly performant, yet faster and less error prone than coding in C, C++, or Java.

I was convinced that my stint at Jane Street would be short and the code I was writing was all throwaway, so I made a choice to maximize my own productivity without worrying about whether others could use the code later. Six months and 80,000 lines of code later, I realized I was wrong: I took a full-time position at Jane Street and soon started hiring to create a research group there.

At this time, the firm was casting around for a new approach to building software. The systems that powered the company in its first years were primarily written in VBA and C#. Indeed, the core trading systems were Excel spreadsheets with a great deal of custom VBA code. This was a great way to get up and running quickly, but it was clear from the start that this was not a sustainable approach.

In 2003, Jane Street began a rewrite of its core trading systems in Java. The rewrite was eventually abandoned, in part because the resulting code was too difficult to read and reason about—far more difficult, indeed, than the VBA that was being replaced. A big part of this was Java's verbosity, but it was

more than that. The VBA code was written in a terse, straight-ahead style that was fairly easy to follow. But somehow when coding in Java we built up a nest of classes that left people scratching their heads when they wanted to understand just what piece of code was actually being invoked when a given method was called. Code that made heavy use of inheritance was particularly difficult to think about, in part because of the way that inheritance ducks under abstraction boundaries.

In 2005, emboldened by the success of the research group, Jane Street initiated another rewrite of its core trading systems, this time in OCaml. The first prototype was done in three months, and was up and trading three months after that. The use of OCaml in the company has only expanded since then. Today it is used to solve problems in every part of the company, from accounting to systems administration, and that effort continues to grow. In recent years, the trading side of the firm has increased its use of the language, and OCaml training is now a standard part of the curriculum for new trading hires. Overall, the transition to OCaml has been a huge success, resulting in far stronger technology than we could have achieved otherwise.

Why OCaml?

What is it about the language that makes it work so well? Here is a short summary of what I perceive as OCaml's key strengths.

▸ **Concision.** Our experience with OCaml on the research side convinced us that we could build smaller, simpler, easier-to-understand systems in OCaml than we could in languages such as Java or C#. For an organization that valued readability, this was a huge win.

▸ **Bug detection.** Programmers who are new to OCaml are often taken aback by the degree to which the type system catches bugs. The impression you get is that once you manage to get the typechecker to approve of your code, there are no bugs left. This isn't really true, of course; OCaml's type system is helpless against many bugs. There is, however, a surprisingly wide swath of bugs against which the type system is effective, including many bugs that are quite hard to get at through testing.

▸ **Performance.** We found that OCaml's performance was on par with or better than Java's, and within spitting distance of languages such as C or C++. In addition to having a high-quality code generator, OCaml has an incremental GC (garbage collector). This means the GC can be tuned to do small chunks of work at a time, making it more suitable for soft real-time appli-

Figure 1. Expression type and evaluator in OCaml.

```
type 'a expr = | True
               | False
               | And  of 'a expr * 'a expr
               | Or   of 'a expr * 'a expr
               | Not  of 'a expr
               | Base of 'a

let rec eval eval_base expr =
   let eval' x = eval eval_base x in
   match expr with
   | True  -> true
   | False -> false
   | Base base -> eval_base base
   | And (x,y) -> eval' x && eval' y
   | Or (x,y)  -> eval' x || eval' y
   | Not x     -> not (eval' x)
```

Figure 2. Expression type and evaluator in Java.

```
public abstract class Expr<T> {

  public interface Evaluator<T> { boolean evaluate(T value); }
  public abstract boolean eval(Evaluator<T> evaluator);

  public class True<T> extends Expr<T> {
    public boolean eval(Evaluator<T> evaluator) { return true; }
  }
  public class False<T> extends Expr<T> {
    public boolean eval(Evaluator<T> evaluator) { return false; }
  }
  public class Base<T> extends Expr<T> {
    public final T value;
    public Base(T value) { this.value = value; }
    public boolean eval(Evaluator<T> evaluator)
    { return evaluator.evaluate(value); }
  }
  public class And<T> extends Expr<T> {
    public final Expr<T> expr1;
    public final Expr<T> expr2;
    public And(Expr<T> expr1, Expr<T> expr2) {
      this.expr1 = expr1;
      this.expr2 = expr2;
    }
    public boolean eval(Evaluator<T> evaluator) {
      return expr1.eval(evaluator) && expr2.eval(evaluator);
    }
  }
  public class Or<T> extends Expr<T> {
    public final Expr<T> expr1;
    public final Expr<T> expr2;
    public Or(Expr<T> expr1, Expr<T> expr2) {
      this.expr1 = expr1;
      this.expr2 = expr2;
    }
    public boolean eval(Evaluator<T> evaluator) {
      return expr1.eval(evaluator) || expr2.eval(evaluator);
    }
  }
  public class Not<T> extends Expr<T> {
    public final Expr<T> expr;
    public Not(Expr<T> expr) { this.expr = expr; }
    public boolean eval(Evaluator<T> evaluator)
    { return !expr.eval(evaluator); }
  }
}
```

cations such as electronic trading.

▸ **Pure, mostly.** Despite how functional programmers often talk about it, mutable state is a fundamental part of programming, and one that cannot and should not be done away with. Sending a network packet or writing to disk are examples of mutability. A complete commitment to immutability is a commitment to never building anything real.

Mutable state has its costs, however. Mutation-free code is generally easier to reason about, making interactions and dependencies between different parts of your codebase explicit and easier to manage. OCaml strikes a good balance here, making mutation easy, but making immutable data structures the default. A well-written OCaml system almost always has mutable state, but that state is carefully limited.

Perhaps the easiest of these advantages to demonstrate concretely is that of concision. The importance of concision is clear: other things being equal, shorter code is easier to read, easier to write, and easier to maintain. There are, of course, limits: no good is done by reducing all your function names to single characters, but brevity is nonetheless important, and OCaml does a lot to help keep the codebase small.

One advantage OCaml brings to the table is type inference, which obviates the need for many type declarations. This leaves you with code that is roughly as compact as code written in dynamic languages such as Python or Ruby. At the same time, you get the performance and correctness benefits of static types.

Consider the following OCaml function map for transforming the elements of a tuple.

```
let map f (x,y,z) =
    (f x, f y, f z)
```

Here, `map` is defined as a function with two arguments: a function `f` and a triple `(x,y,z)`. Note that `f x` is the syntax for applying the function `f` to `x`.

Now consider what this would look like in C# 4.0:

```
Tuple<U,U,U> Map<T,U>
  (Func <T,U> f, Tuple<T,T,T> t)
{
  return new Tuple<U,U,U>
   (f(t.item1), f(t.item2), f(t.item3));
}
```

The C# code, while functionally equivalent, looks cluttered, with the real structure obscured by syntactic noise.

Another source of concision is OCaml's notation for describing types. At the heart of that notation is the notion of an *algebraic datatype*. Algebraic datatypes are what you get when you have a system that includes two ways of building up new types: products and sums.

A *product type* is the more familiar of the two. Tuples, records, structs, and objects are all examples of product types. A product type combines multiple values of different types into a single value. These are called product types because they correspond mathematically to Cartesian products of the constituent types.

A *sum type* corresponds to a disjoint union of the constituent types, and it is used to express multiple possibilities. Where product types are used when you have multiple things at the same time (*a* and *b* and *c*), sum types are used when you want to enumerate different possibilities (*a* or *b* or *c*). Sum types can be simulated (albeit somewhat clumsily) in object-oriented languages such as Java using subclasses, and they show up as union types in C. But the support in the type systems of most languages for interacting with sum types in a safe way is surprisingly weak.

Figure 1 provides an example of algebraic datatypes at work. The code defines a type for representing Boolean expressions over a set of base predicates and a function for evaluating those expressions. The code is generic over the set of base predicates, so the subject of these expressions could be anything from integer inequalities to the settings of compiler flags.

The sum type `expr` is indicated by the pipes separating the different arms of the declaration. Some of those arms, such as `True` and `False`, are simple tags, not materially different from the elements of an enumeration in Java or C. Others, such as `And` and

Figure 3. Destuttering a list.

```
# Removes sequential duplicates, e.g.,
# destutter([1,1,4,3,3,2]) = [1,4,3,2]

def destutter (list):
  l = []
  for i in range(len(list)):
    if list [i] != list[i+1]:
      l.append(list[i])
  return l
```

This code looks pretty straightforward, but it has a bug: it doesn't properly handle the end of the list. Here's one way of fixing it:

```
def destutter (list):
  l = []
  for i in range(len(list)):
    if i + 1 >= len(list) or list[i] != list[i+1]:
      l.append(list[i])
  return l
```

Now let's see what happens when writing more or less the same function in OCaml, with more or less the same bug:

```
let rec destutter l=
 match l with
 | []          -> []
 | x :: y :: rest ->
  if x = y then destutter (y :: rest)
  else x :: destutter (y :: rest)
```

Figure 4. Connection types.

```
type connection_state =
| Connecting
| Connected
| Disconnected

type connection_info = {
  state:              connection_state;
  server:             inet_addr;
  last_ping_time:     time option;
  last_ping_id:       int    option;
  session_id:         string option;
  when_initiated:     time option;
  when_disconnected:  time option;
}
```

`Not`, have associated data, and that data varies between the cases. This type actually contains both sums and products, with the `And` and `Or` branches containing tuples. Types consisting of layered combinations of products and sums are a common and powerful idiom in OCaml.

One notable bit of syntax is the type variable **'a**. A type variable can be instantiated with any type, and this is what allows the code to be generic over the set of base predicates. This is similar to how generic types are handled in Java or C#. Thus, Java's `<A>List` would be rendered as `'a list` in OCaml.

The function `eval` takes two arguments: `expr`, the expression to be evaluated; and `eval _ base`, a function for evaluating base predicates. The code is generic in the sense that `eval` could be used for expressions over any type of base predicate, but `eval _ base` must be provided in order to evaluate the truth or falsehood of those base predicates. The function `eval'` is defined as shorthand for invoking recursive calls to `eval` with `eval _ base` as an argument. Finally, the `match` statement is used for doing a case analysis of the possible structures of the expression, calling out to `eval _ base` when evaluating a base predicate, and otherwise acting as a straightforward recursion over the structure of the datatypes.

Figure 2 shows how the same code might be rendered in Java. The verbosity is immediately striking. Adding a single case such as `And` takes two (short) lines in OCaml and eight in Java—and the Java code is actually pretty minimal as these things go. If you wanted to allow the creation of other algorithms around this expression type that are not baked into the class definition, then you probably want to use the visitor pattern, which will inflate the line count considerably.

Another facet of the language that demands some further explanation is the ability of the type system to catch bugs. People who are not familiar with OCaml and related languages (and some who are) often make the mistake of underestimating the power of the type system. It is easy to conclude that all the type system does for you is ensure you passed in your parameters correctly (for example, that you provided a float where you were supposed to provide a float).

But there is more to it than that. Even naive use of the type system is eerily good at catching bugs. Consider the Python code for destuttering a list (that is, removing sequential duplicates) as shown in Figure 3. It uses OCaml's pattern-matching syntax to get access to the elements of the list. Here `::` is the list constructor, and `[]` indicates an empty list. Thus, the `[]` case matches the empty list, and the `x::y::rest` case matches lists that have at least two elements, `x` and `y`. The variable `rest` refers to the (potentially empty) remainder of the list.

Like the Python example, this code fails to contemplate what happens when you get to the end of the list and have only one element left. In this case, however, you find out about the problem not at runtime but at compile time. The compiler gives the following error:

```
File "destutter.ml", line 2,
characters 2-125:
Warning 8: this pattern-
matching is not exhaustive.
Here is an example of a val-
ue that is not matched:
_ ::[]
```

The missing case, `_ ::[]`, is a list with a single element.

You can fix the code (and satisfy the compiler) by adding a handler for the missing case:

```
let rec destutter l =
  match l with
  | []                  -> []
  | x :: []             -> x :: []
  | x :: y :: rest ->
     if x = y then destutter (y
     :: rest)
     else x :: destutter (y ::
     rest)
```

The error here is a trivial one that would be found easily by testing. But the type system does just as well in exposing errors that are hard to test, either because they show up only in odd corner cases that are easy to miss in testing, or because they show up in complex systems that are difficult to mock up and exercise exhaustively.

Straight out of the box, OCaml is pretty good at catching bugs, but it can do even more if you design your types carefully. Consider as an example the following types for representing the state of a network connection as illustrated in Figure 4.

Figure 5. Connection types revisited.

```
type connecting   = { when_initiated: time; }
type connected    = { last_ping : (time * int) option;
                      session_id : string; }
type disconnected = { when_disconnected: time; }

type connection_state =
| Connecting    of connecting
| Connected     of connected
| Disconnected of disconnected

type connection_info = {
  state:  connection_state;
  server: inet_addr;
}
```

The `connection_state` type is a simple enumeration of three named states that the connection can be in; `connection_info` is a record type containing a number of fields describing different aspects of a connection. Note that the fields that have option at the end of the type are essentially nullable fields. (By default, values in OCaml are guaranteed to be non-null). Otherwise, there is nothing about this code that is all that different from what you might write in Java or C#.

Here is some information on the individual record fields and how they relate to each other:

- `server` indicates the identity of the server on the other side of the connection.
- `last_ping_time` and `last_ping_id` are intended to be used as part of a keep-alive protocol. Note that either both of those fields should be present, or neither of them should. Also, they should be present only when state is `Connected`.
- The `session_id` is a unique identifier that is chosen afresh every time the connection is reestablished. It also should be present only when state is `Connected`.
- `when_initiated` is for keeping track of when the attempt to start the connection began, which can be used to determine when the attempt to connect should be abandoned. This should be present only when state is `Connecting`.
- `when_disconnected` keeps track of when the connection entered the `Disconnected` state, and should be present only in that state.

As you can see, a number of invariants tie the different record fields together. Maintaining such invariants takes real work. You need to document them carefully so you do not trip over them later; you need to write tests to verify the invariants; and you must exercise continuing caution not to break the invariants as the code evolves.

But we can do better. The rewrite in Figure 5 uses a combination of product and sum types that more precisely represents the set of allowable states of a connection. In particular, there is a different record type for each of the three states, each containing the information that is relevant just to that state. Information that is always relevant (in this case, just the `server`) is pushed to the top-level record. Also, we have made it explicit that `last_ping_time` and `last_ping_id` are either both present or both absent by representing them as `last_ping`, which is an optional pair.

By doing all of this, we have embedded into the type many of the required invariants. Now that the invariants are part of the types, the compiler can detect and reject code that would violate these invariants. This is both less work and more reliable than maintaining such invariants by hand.

This example uses algebraic datatypes to encode invariants, but OCaml has other tools for doing the same. OCaml's module system is one example, allowing you to specify invariants in the interface of a module. Unlike most object-oriented languages, it is possible to express complex joint invariants over multiple different types. More generally, OCaml's modules are a powerful tool for breaking down a codebase into small, understandable pieces, where the interactions between those pieces is under the programmer's explicit control.

The type system's ability to catch bugs is valuable even for small solitary projects, but it truly shines in a collaborative environment where multiple developers work together on a long-lived codebase. In addition to finding bugs, type signatures play a surprisingly valuable role as a kind of guaranteed-to-be-correct documentation. In the context of an evolving codebase, invariants enforced by the type system have the benefit of being more durable than those enforced by convention, in that they are less likely to be broken accidentally by another developer.

Limitations

None of this is to say that OCaml is without its flaws. There are, of course, all of the problems associated with being a minority language. OCaml has a great community that has generated a rich set of libraries, but that collection of libraries pales in comparison with what is available for Python, C, or Java. Similarly, development tools such as IDEs, profilers, and debuggers are there, but are considerably less mature and feature-full than their cousins in more mainstream languages.

Another limitation of OCaml has to do with parallelism. The OCaml runtime has a single runtime lock, which means that one must use multiple processes to take advantage of multiple cores on a single machine. For the most part, this fits our development model well: we prefer message passing to shared-memory threads as a programming model for parallelism, since it leads to code that is easier to reason about and it scales better to systems that cross multiple physical machines. The tools available in the wider OCaml world for doing this kind of multiprocess programming, however, are still maturing.

But OCaml's limitations are not fundamental in nature. They have more to do with the details of the implementation or the popularity of the language and not with the language itself. In the end, that is what I find most puzzling. I am now quite convinced that the core ideas behind OCaml are enormously valuable, as evidenced by the fact that OCaml itself, whatever its limitations, is a profoundly effective and powerful tool. Yet, those ideas remain stubbornly outside of the mainstream.

Perhaps this is finally on the verge of changing. Languages such as F# and Scala are bringing some of the ideas behind OCaml and Haskell to a wider audience by integrating themselves within the Dotnet and Java ecosystems, respectively. Maybe 10 years from now, we will no longer need to ask why these ideas have failed to catch on in the wider world. But there is no reason to wait. You can add OCaml to your toolbox now. C

Related articles on queue.acm.org

A Conversation with Arthur Whitney
http://queue.acm.org/detail.cfm?id=1531242

Passing a Language through the Eye of a Needle
Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes
http://queue.acm.org/detail.cfm?id=1983083

The Next Big Thing
George Neville-Neil
http://queue.acm.org/detail.cfm?id=1317398

Yaron Minsky joined Jane Street in 2003, where he founded the quantitative research group. Since 2007 he has managed the firms's technology group.

26th IEEE INTERNATIONAL PARALLEL and DISTRIBUTED PROCESSING SYMPOSIUM

Sponsored by IEEE computer society Technical Committee on Parallel Processing

In cooperation with ACM SIGARCH, IEEE Computer Society Technical Committee on Computer Architecture, and IEEE Computer Society Technical Committee on Distributed Processing

IPDPS 2012 SHANGHAI

Shanghai, China
21 May - 25 May 2012

Hosted by 上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

www.ipdps.org

GENERAL CO-CHAIRS

Minglu Li (Shanghai Jiao Tong University, China)
Cho-Li Wang (University of Hong Kong, China)

PROGRAM CO-CHAIRS

Leonid Oliker
(Lawrence Berkeley National Laboratory, USA)
Katherine Yelick
(Lawrence Berkeley National Lab & University of California at Berkeley, USA)

WORKSHOPS CHAIR

Ümit V. Çatalyürek (Ohio State University, USA)

PhD FORUM CO-CHAIRS

Luc Bougé (ENS Cachan, France)
Bo Hong (Georgia Institute of Technology, USA)

PhD Forum: Held in the middle of the week, this event offers graduate students an opportunity to present a research poster describing their ongoing dissertation to the entire IPDPS conference audience and to submit a short paper for publication in the proceedings of the IPDPS Workshops. The deadline for submissions is January 11, 2012. See details at www.ipdps.org.

IMPORTANT DATES

1 October 2011	*Call for Papers closed:* Regular Conference
November 2011-February 2012	*Call for Papers Due Dates:* Workshops (see www.ipdps.org)
19 December 2011	*Author notification:* Regular conference papers
11 January 2012	PhD Forum submissions due
1 February 2012	*Camera-ready due:* Regular conference papers
21 February 2012	*Camera-ready due:* Workshops & PhD Forum

IPDPS 2012 WORKSHOPS

IPDPS workshops, held on the first and last day of the conference, are a major part of the IPDPS week-long family of events. Below is the list of workshops planned for IPDPS 2012 in Shanghai. These workshops provide the IPDPS community an opportunity to explore special topics, and the goal of the workshops is to present work that is more preliminary and cutting-edge or that has more practical content than the more mature research presented in the main symposium. Each workshop has its own requirements and schedule for submissions and all are linked from the workshops page on the IPDPS Website. Proceedings of the workshops are published by the IEEE Digital Library and are distributed at the conference. Note that most workshops have a due date for submissions later than the author notification date for regular conference papers.

HCW	Heterogeneity in Computing Workshop
RAW	Reconfigurable Architectures Workshop
HIPS	High-Level Parallel Programming Models & Supportive Environments
NIDISC	Nature Inspired Distributed Computing
HiCOMB	High Performance Computational Biology
APDCM	Advances in Parallel & Distributed Computing Models
CASS	Communication Architecture for Scalable Systems
HPPAC	High-Performance, Power-Aware Computing
HPGC	High-Performance Grid and Cloud Computing
SMTPS	System Management Techniques, Processes, & Services
EduPar	NSF/TCPP Workshop on Parallel and Distributed Computing Education
PDSEC	Parallel & Distributed Scientific & Engineering Computing
DPDNS	Dependable Parallel, Distributed and Network-Centric Systems
HOTP2P	Hot Topics in Peer-to-Peer Systems
MTAAP	Multi-Threaded Architectures and Applications
LSPP	Large-Scale Parallel Processing
PCO	Parallel Computing and Optimization
ASHES	Accelerators and Hybrid Exascale Systems
ParLearning	Parallel and Distributed Computing for Machine Learning and Inference Problems
HPDIC	High Performance Data Intensive Computing
CloudFlow	Workflow Models, Systems, Services and Applications in the Cloud
JSSPP	Job Scheduling Strategies for Parallel Processing
LSDSS	Large Scale Distributed Service-oriented Systems
PLC	Multicore and GPU Programming Models, Languages and Compilers
STDN	Security and Trust of Distributed Networking Systems

DOI:10.1145/2018396.2018414

Users will speak rather than type, watch video rather than read, and use technology socially rather than alone.

BY MARTI A. HEARST

'Natural' Search User Interfaces

WHAT DOES THE future hold for search interfaces for users? Today's familiar Web search interface works well for tens of millions of people and billions of queries a year, but few innovations in search interfaces gain wide-enough acceptance to replace the standard type-keywords-in-entry-form/view-results-in-a-vertical-results-list interface. This is partly because search is a means toward another end, and reading text is a mentally demanding task. The fewer distractions while reading, the more usable the interface. Additionally, search, like email, is used by nearly everyone using the Web, so its features and functions must be understandable to an enormous and diverse population.[13]

Future trends in search interfaces will most likely reflect trends in the use of IT generally. Today, there is a notable trend toward more "natural" user interfaces: pointing with fingers rather than mice, speaking rather than typing, viewing videos rather than reading text, and writing full sentences rather than artificial keywords. (The term "natural interface" is promoted by researchers at Microsoft, among others.) Not surprisingly, people are drawn to interfaces that allow them to think and move in a manner like what they use in their non-computing lives, but only recently has technology been able to support it.

There is also a trend toward social rather than solo use of IT, with these multi-person interactions often recorded, stored, and indexed for later viewing. Again, many people would have preferred non-isolated computer use from the start, but technology and user-interface design did not support it well until recently.

Technology is advancing toward integration of massive quantities of user behavior and large-scale human-generated knowledge bases. Search today benefits from the tracking of search behavior over hundreds of millions of queries to improve ranking, offer accurate spelling suggestions, auto-suggest query terms in real time as the user types, and suggest concepts related to a query. Integration with databases and more sophisticated processing place search at the cusp of being able to support smarter, data-driven, focused interfaces for advanced search.

These trends are, or will be, interweaving in various ways, with interesting ramifications for search interfaces

» key insights

- **"Natural" modes of interaction are starting to be commonplace in hardware and software tools, influencing search interfaces in interesting ways.**
- **These changes lend urgency to the research problems of analyzing video content, interpreting spoken and written natural language, and supporting collaboration among people seeking information together.**
- **Content analysis over huge collections of user behavior data, combined with interactive user-interface design could lead to breakthroughs in such long-standing problems as human-computer dialogues for question answering.**

ILLUSTRATION BY GARY MARSH


FFT

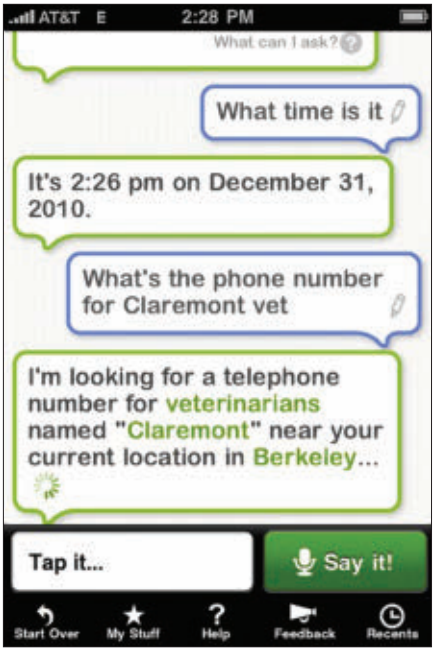


Figure 1. The Siri interface accepts speech as input, attempting to support a dialogue; in the first action, a query for a phone number shows a message conveying the system's understanding of the question as it performs a search.

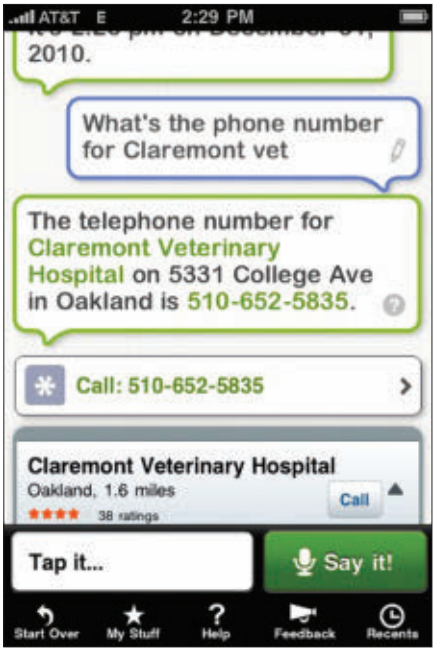


Figure 2. In the second action, the system replaces the confirmation bubble with the answer, offering a command to make the phone call; note the use of graphics alongside text to enrich the output.

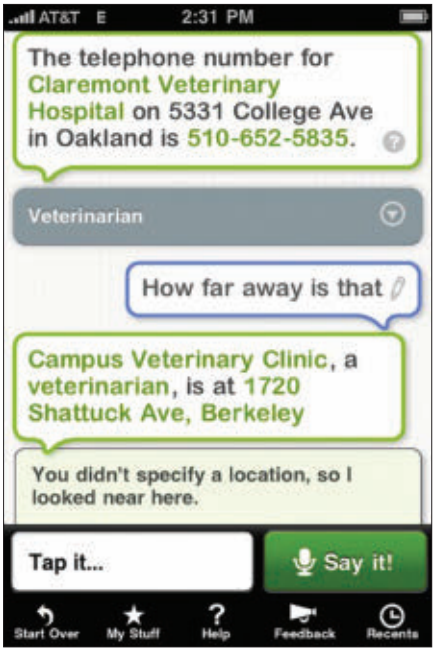


Figure 3. In the concluding action, the system partly understands the intention of the question (about a location) but incorrectly interprets a request for a distance as a request for a location.

and suggesting promising directions for research.

Speech Input

Speech-based user interfaces generally, and speech for search input in particular, are likely to gain a much stronger presence in the coming years. At least three technological trends support the move toward spoken queries: First, phone-based mobile devices provide a natural way to capture speech, since phones are used in large part for spoken conversations. Second, the technology for speech recognition, after years of only incremental progress, is improving by leaps and bounds, thanks to huge data repositories being generated through the use of mobile phones. (To assemble a large training set of spoken corrected data for its speech-recognition system, Google hosted, from 2007 to 2010, a free 411 information service for phones.[28]) And third, touch-screen interfaces are increasingly popular, especially when paired with mobile devices. Neither small devices nor touch screens lend themselves well to typing, making spoken input more attractive, though clever finger-swipe-based input methods (such as ShapeWriter for entering text[39] and Gesture Search for menu navigation[19]) provide compelling alternatives to typing.

These trends suggest voice-activated queries and commands are likely to increase rapidly in the next few years as response time and accuracy continue to improve.

The next likely development following on voice-based input is a dialogue-like give and take. Though not yet a reality, recent advances are bringing closer the dream of an intelligent interactive agent. For example, the Siri system provides an interface combining local information, speech recognition, easy editing of voice recognition, and visual display of search results. Siri, which was acquired by Apple in April 2010, originated from a Defense Advanced Research Projects Agency research project called CALO (http://www.ai.sri.com/project/CALO), in which dozens of computer-science researchers developed machine learning, reasoning, knowledge bases, and other technology to create an intelligent personal assistant.[4,35]

Though the user's ability to accurately follow up one request with another is limited in Siri, good interface design helps bridge the gap in the back end, since the user sees alternatives and is able to make corrections (see figures 1–3). Note that Siri also attempts to use searchers' contextual information, including current location. Enormous research interest[5,20] and commercial development focuses on using time, location, and other contextual cues for search and related applications, and will continue to increase in importance, especially for mobile platforms.

Voice input also has drawbacks, the most significant being that speaking makes noise and can disturb people around the speaker. An exciting research advance would be a microphone that uptakes the words the speaker says but somehow prevents those around the speaker from hearing the words, like a science fiction "cone of silence." Such an invention would have wide-ranging utility for mobile phones.

Social Search: Collaboration

Though observational studies have found that people often search collaboratively, tools have only recently been developed to explicitly support people searching together. Such support reflects a broader research renaissance in tools for real-time shared activity (such as shared online whiteboards and document-editing tools).

One exciting development in collaborative search, from Pickens et al.,[11,29] assumes the ranking algorithm should allow users to work at their own pace but be influenced in real time by their teammates' search activities. The searchers should not step on one another's proverbial toes; if one person issues a new query, others' thoughts should not be interrupted.

Pickens et al.[11,29] addressed this issue by developing an algorithm that combines multiple rounds of queries from multiple searchers during a single search session (see Figure 4), using two criteria for weighting results—both functions of the ranked list of documents returned for a given query. The first variable is "freshness," which is higher for documents not yet viewed, while "relevance" is higher for documents closely matching the query. These two factors are combined and continuously updated based on new queries and searcher-specified relevance judgments.

In addition, Pickens et al.[11,29] assigned different roles to the members of a team. For example, the "Prospector" is in charge of creating new queries to explore new parts of the information space, and the "Miner" looks at the retrieved results to determine which are relevant. Documents not yet looked at are queued up for the Miner interface according to freshness/relevance weighted scores. The Prospector is shown new query-term suggestions based on how they differ from queries already issued, as well as on the relevance judgments made by the Miner. Each role has its own interface; a third view is used to show continually updating information about the queries that have been issued, the documents that have been marked as relevant, and the system-suggested query terms based on the actions of the users.

Another approach to supporting real-time search collaboration, described by Jetter et al.,[16] used a large work surface and input devices combining physical manual manipulation with virtual markings. The interface was evaluated on a complex collaborative search task, that of a group of people selecting a product, where each member of the group has different preferences that act as constraints (such as when choosing a hotel, one needs a heated pool, another wants one that received at least four stars of recommendation, and a third wants the price below a certain amount). Jetter et al.'s solution used a combination of faceted navigation[37] and filter-flow visualization,[38] showing how many constraints are met by a set of items, given certain constraints. The visualization was displayed on a shared horizontal workspace, where the controls were manipulated through physical selectors (see Figure 5). Collaboration was facilitated by allowing each user to work privately on a corner of the workspace, then let the results from each piece of the query flow into the rest of the group's query specification. A careful usability study by Jetter et al. found this approach produced results

Figure 4. Collaborating on a video-search task using technology developed by Pickens et al.[29]; each user views a different unique interface, as well as a shared view. The results of one person's work change the rank ordering of what is seen by the other person.

Figure 5. A collaborative search-formulation tool making use of a large table display, physical input devices, and visualization; from Jetter et al.[16]

as good as those using a standard Web-based faceted navigation interface but with more bonhomie among the collaborators.

Social Search: Asking Other People

Research suggests that much online interaction on social sites is for the social experience of the interaction, rather than for problem-centric information seeking.[12] Reflecting this, a study by Morris et al.[24] found the questions asked of others via social networks do not necessarily involve the kinds of information found on static Web pages. Morris et al. asked survey respondents to supply questions they had posed to their social networks on Twitter and Facebook, manually classifying the 249 examples and finding only 17% were for factual information one would typically seek from a Web page (such as how to, say, put an Excel file into LaTex). The most common categories were requests for recommendations (29%), opinions (22%), rhetorical questions (14%), requests for others to join social events (9%), favors (4%), and social connections, including job openings (3%) and offers of various kinds (1%).

A study of the Aardvark expert social-question-answering system (http://www.vark.com) found similar results, with 65% of a random sample of 1,000 queries reflecting a subjective attitude.[15] The questions asked on the social-question-answering site Quora also tend to be subjective and opinion-based; for instance, "What does Dustin Moskovitz think of the new Facebook movie?" was answered by the subject of the question himself.

Unclear is what the best user interfaces are for representing this more social kind of search. Freyne et al.[10] conducted a small study in which different kinds of social cues were shown via icons alongside search-results listings. Subjective results showed a positive preference toward cues showing which articles were read frequently or annotated by others. Yahoo experimented (2005–2009) with the MyWeb system in which search results were augmented with an avatar of the person in the user's social network who had recommended the page, along with the recommendation. In March 2011, Google introduced a social-search tool called "+1" with a similar interface. Significant experimentation on incorporating social information into search results listings is likely over the next few years.

When using a social network to try to answer questions, especially in a work situation, research is ongoing about how best to distribute the related information needs among experts, either within an organization or across the Internet generally.[18,21] Recent work by Richardson and White[34] deployed and studied an instant-messaging-based question-answering service that matched the asker's questions against predefined profiles of more than 2,000 potential answerers' expertise, based on their availability. The system contacted three experts at a time, in descending order of how well their profiles matched the content of the question. If an offer to answer was not received within a fixed time limit, the request was sent to a wider circle of experts. If an answerer accepted a request, the other outstanding requests were cancelled. The tool then mediated the conversation between questioner and answerer, asking questioners to rate their satisfaction with the answer.

Richardson and White[34] examined log data for this system to form an interruption cost model, including how many people should be sent a question in order to minimize disruption while maximizing the likelihood of receiving an informed answer, whether a question will be answered, and how well the asker will be satisfied with the answer received.

Expert solicitation systems that are sophisticated about targeting people with the right expertise and state of mind to address a request are likely to become a fixture in knowledge-centric workplaces, as well as in volunteer causes (such as the Peer2Patent project for community input of patent prior art[26]).

Social Search: Crowdsourcing

The word "collaboration" as it is used here refers to a set of people working together closely, usually synchronously, to achieve a goal. "Crowdsourcing" refers to large groups of people not necessarily working together knowingly but each contributing in small ways, leading to a greater whole, as seen, in, for example, Wikipedia editing.

Crowdsourcing in information seeking is seen in Web sites in which communities curate and rate information and share it with others, including question-answering sites, and in product-reviewing sites, bookmark-sharing sites like Delicious, and news-ranking and aggregation sites like Digg. The more-explicitly networked social tools (such as Twitter and Facebook) also function as real-time socially targeted information sources.

Multiple efforts have sought to use explicit user input to improve search-results ranking, though few survive; for instance, Google's SearchWiki, which allowed users to explicitly reorder search results and share this re-ranking information with others, was shut down in 2010. The Blekko Web search engine, launched October 2010, is an attempt to use sophisticated algorithms combined with community curation to improve results rankings; its founder also started the Open Directory Project, a crowdsourced yellow pages for the Web. With Blekko, users can create "vertical," or subject-specific, search by labeling Web pages with a category label preceded by a slash; they can also mark pages as spam. These two operations together impose crowdsourced quality control over retrieved Web pages. Blekko also provides a social feature allowing users to see if their friends have marked particular pages with a "/like" slashtag. It remains to be seen if explicit crowdsourcing will scale for search results ranking.

Crowdsourcing usually refers to people explicitly contributing to an effort, but Web search engines have used a form of implicit crowdsourcing for years, by modifying ranking algorithms based on huge quantities of user clickthrough data[17] or predicting which vertical subject area (such as music, news, and travel) to use to augment a query.[7] Richer user behavior data (such as mouse movements, page dwell time, and searchers' click paths many steps from the search results page—even across domains—to their destination page) has helped produce useful suggestions of pages not related to the original page through close keyword matches.[36]

Natural Language-like Queries

Though keyword querying remains standard practice on the Web, savvy users have been typing more detailed queries for years, and Web search engines have greatly improved their ability to handle long queries. Research has shown that people prefer natural expression of queries over keywords,[3,30] and Web search engine query length continues to increase. According to Experian Hitwise,[22] a global online competitive intelligence service, when comparing queries over a four-week period (August–September 2010) to the same four-week period in 2009, found that searches of from five to eight words were up 10%, while searches of from one to four words were down 2%. The growth of query length suggests a desire to express one's information needs more thoroughly and may pave the way toward full-sentence queries. Spoken queries are also likely to be full sentences when speech recognition is faster and more accurate.

Longer queries are also being helped by the online use of colloquial language. When most content is technical or scientific (as was characteristic of the early Web), there is less likely an easy-to-find match between a lay user's words and the words used in the informative documents. Popular question-answering sites (such as Answers.com, Quora, and Yahoo Answers) that store user-generated content bridge colloquial and formal language directly in relevant documents; for example, if a searcher needs a device to connect both a Wii and a DVD player to a TV, but does not know what that device is called, a keyword query could fail. But the query "how do I connect wii and dvd to my tv" turns up a nearly perfect match on a question-answering site, with the solution being a product called either "video selector" or "two-way A/V switcher." The point is that, though the searcher lacks the vocabulary to look up what is needed, the searcher has the same vocabulary as other people in the same cognitive situation. The combination of text worded colloquially and search engines that do a good job with sentence-length queries helps resolve the vocabulary problem. Considerable work has focused on how to search question-answering sites[1,2]; ranking algorithms that make use of these mappings will continue to improve results for difficult queries.

Though observational studies have found that people often search collaboratively, tools have only recently been developed to explicitly support people searching together.

Another technical development that may help users who express themselves through long queries is systems that support quasi-natural language interfaces. The new syntax is tolerant of variations, relatively robust, and "exhibit[s] slight touches of natural language flexibility."[25] These interfaces are seen in Web search engines supporting various wordings for certain kinds of questions that retrieve answers from a database, as in "Istanbul time," "What is the time in Istanbul?," and "What time is it? Istanbul." Blekko allows query modification through a simple slash notation to refine results to predefined categories (such as "istanbul /tech" for search results about technology and "istanbul /people" for results labeled relevant to people).

Miller et al.[23] developed tools for "sloppy commands," meaning users have a lot of flexibility as to how they express the command, so memorization is not required to make use of them. The "linguistic command line" of Enso (later Ubiquity)[8,33] experimented with leniency in operating system command lines. The Quicksilver application lookup tool for Apple operating systems supports a hybrid command/GUI interface, using continuous feedback to whittle down the available choices to include what the user has typed so far that still matches available commands.

The Wolfram Alpha search engine provides a range of predefined query types that mix structured forms with some flexibility in word order, along with a knowledge base and computational back-end able to handle certain combinations of these inputs. For instance, the query "2 slices of pizza with pepperoni" is decomposed into the base information need (information about pizza) refined by units (slices), the quantity (two), and modifications of the baseline concept (with pepperoni). The result is a table listing calorie and nutrition information. However, the system's interpretive range is limited; the query "recipe for pizza with pepperoni" returns the same measurement information as "pizza with pep-

peroni" instead of a recipe.

This hybrid of improved language analysis, command languages making use of structured knowledge bases, and interaction may well lead to more intelligent interfaces and expanded dialogue-like interaction, as discussed earlier regarding the Siri system. The IBM Watson project, which famously beat the top two human champions in the television game show "Jeopardy!" in February 2011, also employs massive language analysis, knowledge-base analysis, and speech recognition, likely setting the stage for future highly advanced natural-language question-answering systems.[9]

Importance of Video Content

Increasing evidence reflects a preference among ordinary information consumers for video and audio content over textual content. Movies have generally replaced books as cultural touchstones in the U.S. A report by Pew Research included a quote from a media executive saying email messages containing podcasts were opened 20% more often than standard marketing email messages.[32]

Also according to Pew, 52% of U.S. adults have watched online videos, with seven in 10 Internet users saying they have.[31] According to Hitwise, the YouTube video-sharing site was the fifth most visited Web site in the U.S in 2010,[14] and comScore reported in March 2010 that YouTube users generated a greater search volume than Yahoo or Bing.[6]

Video communication is taking some of the trappings of textual communication; for instance, YouTube supports the notion of a video "reply." And when video questions were accepted for the 2008 U.S. presidential primary debates, most citizen-submitted videos selected by the moderators consisted of people pointing the camera at themselves and speaking their question aloud, with a backdrop consisting of a wall in a room in their homes. There were few visual flourishes, and the video did not add much beyond what a questioner in a live audience would have conveyed. Video is fast becoming a conventional way to communicate.

Mobile devices make it easier to capture video, increasing the likelihood of video becoming an even more important form of communication. According to Pew, almost 20% of American adults had, as of 2010, tried video calling on phones or computers, and 23% of U.S. Internet users had used a video chat service (such as Skype). Further, 14% of U.S. Internet users had created and uploaded videos.[31]

Still lacking are truly useful tools for cogently skimming video content, summarizing it in a meaningful way, and, more to the point, searching within and across it, though research is active in this area.

No doubt, the technology to support full video use lags significantly behind that of text, but we can surmise that some handy inventions are not far off. Better tools for quick edits are also likely soon, as they have been for image processing; a popular mobile iPhone app called Instagram allows users to snap a photo with their phones, quickly apply filters to produce an "artsy" look, then immediately share the image with a social network. Instagram claimed it attracted one million users within two months of its introduction, October 2010, and seven million by August 2011.

Still lacking are truly useful tools for cogently skimming video content, summarizing it in a meaningful way, and, more to the point, searching within and across it, though research is active in this area.[27] YouTube provides tools that automatically provide textual closed captioning over spoken language and can also be used for search; so has a startup company called SpeakerText. Faceted navigation[37] has become the method of choice for browsing image collections; perhaps the same will be possible with video collections. However, serious breakthroughs are still needed for both image and video content analysis before such search performance rivals that of text search.

Time constraints imposed by YouTube have resulted in a culture of short videos characterized by focused topics, making title search more effective than it would be if most online videos were longer in duration; for instance, the excellent educational video courses of the Khan Academy (http://www.khanacademy.org) are each shorter than 10 minutes, with subject matter easily browsable by title (as in "Circles: Diameter, Radius, and Circumference" and "Distributive Property of Matrix Products"). But just as search over collections of

books is still not particularly sophisticated, search over movie-length videos may well prove problematic and require alternative approaches.

Conclusion

The future of user interfaces will involve support for natural human interaction, gesturing with fingers, speaking rather than typing, watching video rather than reading, and using IT socially rather than alone. This article has explored why these trends will also affect user interfaces for search, highlighting recent work reflecting these trends. Using advanced processing techniques over huge sets of behavioral data, future search interfaces will better support finding other people to answer questions or provide opinions, more natural dialogue-like interaction, and information expressed as non-textual content through non-textual input. More-natural modes of interaction have long been goals of interface design, but recent developments have brought them closer to reality. C

References

1. Adamic, L.A., Zhang, J., Bakshy, E., and Ackerman, M.S. Knowledge sharing and Yahoo answers: Everyone knows something. In *Proceedings of the 17th International Conference on the World Wide Web* (Beijing). ACM Press, New York, 2008, 665–674.
2. Bian, J., Liu, Y., Agichtein, E., and Zha, H. Finding the right facts in the crowd: Factoid question answering over social media. In *Proceedings of the 17th International Conference on the World Wide Web* (Beijing). ACM Press, New York, 2008, 467–476.
3. Bilal, D. Children's use of the Yahooligans! Web search engine: I. Cognitive, physical, and affective behaviors on fact-based search tasks. *Journal of the American Society of Information Science 51*, 7 (2000), 646–65.
4. Chaudhri, V.K., Cheyer, A., Guili, R., Jarrold, B., Myers, K.L., and Niekrasz, J. A case study in engineering a knowledge base for an intelligent personal assistant. In *Proceedings of the 2006 Semantic Desktop Workshop* (Athens, GA, 2006).
5. Church, K., Neumann, J., Cherubini, M., and Oliver, N. The map trap: An evaluation of map versus text-based interfaces for location-based mobile search services. In *Proceedings of the 19th International Conference on the World Wide Web* (Raleigh, NC, Apr. 26–30). ACM Press, New York, 2010, 261–270.
6. comScore. comScore releases March 2010 U.S. search engine rankings (Mar. 2010); http://www.comscore.com/Press_Events/Press_Releases/2010/4/comScore_Releases_March_2010_U.S._Search_Engine_Rankings
7. Diaz, F. and Arguello, J. Adaptation of offline vertical selection predictions in the presence of user feedback. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development on Information Retrieval* (Boston, July 19–23). ACM Press, New York, 323–330.
8. Erlewine, M.Y. Ubiquity: Designing a multilingual natural language interface. In *Proceedings of the SIGIR Workshop on Information Access in a Multilingual World* (Boston, July 19–23). ACM Press, New York, 2009.
9. Ferrucci, D., Brown, E, Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A.A., Lally, A., Murdock, J.W., Nyberg, E., Prager, J., et al. Building Watson: An overview of the DeepQA Project. *AI Magazine 31*, 3 (2010).
10. Freyne, J., Farzan, R., Brusilovsky, P., Smyth, B., and Coyle, M. Collecting community wisdom: Integrating social search & social navigation. In *Proceedings of the 12th International Conference on Intelligent User Interfaces* (Honolulu, Jan. 28–31). ACM Press, New York, 2007, 52–61.
11. Golovchinsky, G., Shah, C., and Pickens, J. Role-based results redistribution for collaborative information retrieval. *Information Processing and Management 46*, 6 (2010), 773–781.
12. Harper, F.M., Moy, D., and Konstan, J.A. Facts or friends?: Distinguishing informational and conversational questions in social Q&A sites. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems* (Boston, Apr. 4–9). ACM Press, New York, 2009, 759–768.
13. Hearst, M. *Search User Interfaces.* Cambridge University Press, 2009.
14. Hitwise. Facebook was the top search term in 2010 for second straight year (Dec. 29 2010); http://www.hitwise.com/us/press-center/press-releases/facebook-was-the-top-search-term-in-2010-for-sec/
15. Horowitz, D. and Kamvar, S.D. The anatomy of a large-scale social search engine. In *Proceedings of the 19th International Conference on the World Wide Web* (Raleigh, NC, Apr. 26–30). ACM Press, New York, 431–440.
16. Jetter, H.-C., Gerken, J., Zöllner, M., Reiterer, H., and Milic-Frayling, N. Materializing the query with facet-streams: A hybrid surface for collaborative search on tabletops. In *Proceedings of the 29th International Conference on Human Factors in Computing Systems* (Vancouver, Canada, May 7–12). ACM Press, New York, 2011.
17. Joachims, T., Granka, L., Pan, B., Hembrooke, H., and Gay, G. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil, Aug. 15–19). ACM Press, New York, 2005, 154–161.
18. Kautz, H., Selman, B., and Shah, M. Referral Web: Combining social networks and collaborative filtering. *Commun. ACM 40*, 3 (Mar. 1997), 63–65.
19. Li, Y. Gesture search: A tool for fast mobile data access. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology* (New York, Oct. 3–6). ACM Press, New York, 87–96.
20. Liu, C., Rau, P.L.P., and Gao, F. Mobile information search for location-based information. *Computers in Industry 61*, 4 (May 2010), 364–371.
21. Liu, Y. and Agichtein, E. You've got answers: Towards personalized models for predicting success in community question answering. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies* (Columbus, OH, June 15–20). Association for Computational Linguistics, Stroudsburg, PA, 2008, 97–100.
22. McGee, M. The long tail is alive and well. *Small Business Search Marketing* (Sept. 16, 2010); http://www.smallbusinesssem.com/long-tail-alive-well/3659/ and http://twitter.com/Hitwise_US/status/24041444164
23. Miller, R.C., Chou, V.H., Bernstein, M., Little, G., Van Kleek, M., and Karger, D. Inky: A sloppy command line for the Web with rich visual feedback. In *Proceedings of the 21st annual ACM Symposium on User Interface Software and Technology* (Monterey, CA, Oct. 19–22). ACM Press, New York, 2008, 131–140.
24. Morris, M.R., Teevan, J., and Panovich, K. What do people ask their social networks, and why?: A survey study of status message Q&A behavior. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems* (Atlanta, Apr. 10–15). ACM Press, New York, 2010, 1739–1748.
25. Norman, D. The next UI breakthrough: Command lines. *Interactions 14*, 3 (2007), 44–45.
26. Noveck, B.S. Peer to patent: Collective intelligence, open review, and patent reform. *Harvard Journal of Law & Technology 20*, 1 (2006), 123–162.
27. Over, P., Awad, G., Fiscus, J., Antonishek, B., and Michel, M. TRECVID 2010: An introduction to the goals, tasks, data, evaluation mechanisms, and metrics. *Proceedings of the Eighth TRECVID Workshop.* National Institute of Standards and Technology, Gaithersburg, MD, 2010.
28. Peres, J.C. Google wants your phonemes. *InfoWorld* (Oct. 23, 2007); http://www.infoworld.com/t/data-management/google-wants-your-phonemes-539
29. Pickens, J., Golovchinsky, G., Shah, C., Qvarfordt, P., and Back, M. Algorithmic mediation for collaborative exploratory search. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Singapore, July 20–24). ACM Press, New York, 2008, 315–322.
30. Pollock, A. and Hockley, A. What's wrong with Internet searching. *D-Lib Magazine* (Mar. 1997); http://www.dlib.org/dlib/march97/bt/03pollock.html
31. Purcell, K. *The State of Online Video.* Pew Internet & American Life Project, Washington, D.C., June 3, 2010; http://www.pewinternet.org/~/media//Files/Reports/2010/PIP-The-State-of-Online-Video.pdf
32. Ranie, L. *Digital 'Natives' Invade the Workplace.* Pew Internet & American Life Project, Washington, D.C., Sept. 28, 2006; http://pewresearch.org/pubs/70/digital-natives-invade-the-workplace
33. Raskin, A. The linguistic command line. *Interactions 15*, 1 (2008), 19–22.
34. Richardson, M. and White, R. Supporting synchronous social Q&A throughout the question life cycle. In *Proceedings of the 20th International World Wide Web Conference* (Hyderabad, India, Mar. 28–Apr. 1, 2011).
35. Roush, W. The story of Siri, from birth at Sri to acquisition by Apple: Virtual personal assistants go mobile. *Xconomy* (June 2010); http://www.xconomy.com/san-francisco/2010/06/14/the-story-of-siri-from-birth-at-sri-to-acquisition-by-apple-virtual-personal-assistants-go-mobile/
36. White, R.W., Bilenko, M., and Cucerzan, S. Studying the use of popular destinations to enhance Web search interaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Amsterdam, The Netherlands, July 23–27). ACM Press, New York, 2007, 159–166.
37. Yee, K.P., Swearingen, K., Li, K., and Hearst, M. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Fort Lauderdale, FL, Apr. 5–10). ACM Press, New York, 2003, 401–408.
38. Young, D. and Shneiderman, B. A graphical filter/flow representation of Boolean queries: A prototype implementation and evaluation. *Journal of the American Society for Information Science 44*, 6 (July 1993), 327–339.
39. Zhai, S., Kristensson, P.O., Gong, P., Greiner, M., Peng, S.A., Liu, L.M., and Dunnigan, A. Shapewriter on the iPhone: From the laboratory to the real world. In *Proceedings of the 27th International Conference Extended Abstracts on Human factors in Computing Systems* (Boston, Apr. 4–9). ACM Press, New York, 2667–2670.

Marti A. Hearst (hearst@ischool.berkeley.edu) is a professor in the School of Information at the University of California, Berkeley, with an affiliate position in the Computer Science Division. She is the author of *Search User Interfaces*, Cambridge University Press, 2009.

DOI:10.1145/2018396.2018416

Insightful implementers refocus user ambivalence and resistance toward trust and acceptance of new systems.

BY DONGBACK SEO, ALBERT BOONSTRA, AND MARJOLEIN OFFENBEEK

Managing IS Adoption in Ambivalent Groups

PURSUING CONTINUOUS TECHNOLOGY improvement and innovation, organizations of all kinds introduce new systems almost daily, hoping they are adopted by employees, suppliers, customers, and others. From e-reader vendors (such as Amazon and Apple) to insurance companies communicating with their customers through Web applications, the goal is to inspire employees and customers alike to adopt and use new systems. However, despite the best efforts of IS developers, project managers, and trainers, new systems are not always adopted and used as intended. Lacking commitment and constructive feedback from targeted users, IS adoption may simply fail.[8,9]

Here, we explore user behavior toward adopting systems, provide a way to classify users into groups, and suggest strategies for dealing with each such group. Research tends to categorize user behavior in terms of either acceptance[3] or resistance,[6] a view that fails to acknowledge that such behavior covers a range of ambivalence.[12] While people may generally use technology they support, other responses are common as well. For example, Linux workstations have enthusiastic supporters who nevertheless do not use them due to personal or organizational barriers.[11] There are also people who routinely yet only grudgingly use Microsoft Office applications. Such behavior—"supporting but no or low usage" and "resisting but high usage"—is ambivalent. Here, we address how implementers can deal with groups showing ambivalent behavior.

Research findings on IS adoption offer little guidance, with researchers focusing on either people's acceptance by measuring their IS use or use-intentions (such as Ajzen and Fishbein[2] and Davis[3]) or their resistance by measuring their supporting/resisting behaviors (such as Joshi[4]). In doing so, "acceptance" and "resistance" have, implicitly or explicitly, been conceptualized as an either/or proposition, the opposite ends of a single closed dimension.[6] Consequently, user ambivalence remains largely hidden in the literature. Here, we present a case from the Netherlands that highlights two hidden groups of intended users with ambivalent behaviors in adopting an electronic prescription system (EPS) for the country's general medical practitioners (GPs).

» key insights

- **The traditional one-dimensional view of users' system-adoption behavior, from acceptance to resistance, prompted us to explore a more nuanced two-dimensional view, from "high-use versus non-use" to "support versus resistance."**
- **The two-dimensional view highlights inclusion of groups defined by their ambivalent behavior, from "supporting non-users" to "resisting users."**
- **Implementers should aim to develop customized strategies to turn potential users into "supporting and high usage" users of new systems.**

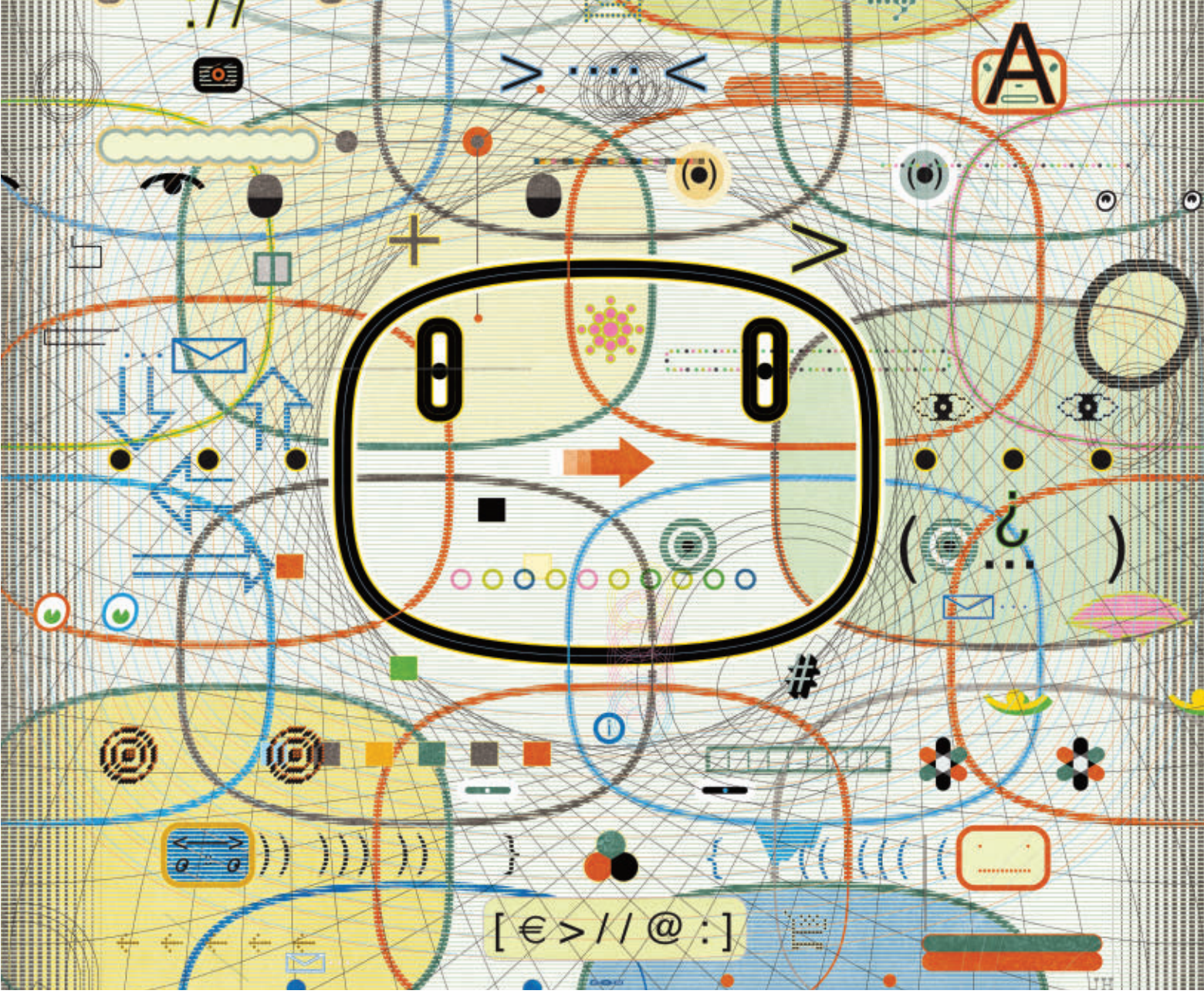


ILLUSTRATION BY JOHN HERSEY

Creating strategies for these ambivalent groups is important, because their behavior is meaningful and differs substantially from groups that are more straightforward in adopting, or not adopting, a system. When supporting non-users are treated as uncooperative mavericks, their initial support for a system could be at risk. And while it seems pragmatic to neglect users' resistance as long as they continue to use the system, the result could be long-term problems. Finally, the groups with ambivalent behavior can influence others' reactions toward the system in unforeseen ways. Such influences suggest implementers need customized strategies to cope with the diverse, sometimes ambivalent, reactions of target groups expected to adopt a new system.

We conceptualize IS acceptance and resistance as two dimensions leading to identification of four types of behavior by user groups. Additionally, we identify two other groups that might not use a new system but are involved in its implementation process through their support of or resistance to its adoption. The resulting model consists of six categories IS implementers might subsequently apply to identify and discuss GP behaviors in adopting the EPS in the Netherlands. Finally, we discuss how implementers can monitor and develop customized strategies, particularly for the two "hidden" groups overlooked in the literature: "resisting users" and "supporting non-users."

Six Actor Groups

Figure 1 includes four user-based quadrants based on two dimensions derived from the IS-acceptance-and-resistance literature.[3,4,6–8,15] The vertical dimension represents interviewed users' IS acceptance measured by their IS use.[3,15] The horizontal dimension represents IS resistance measured through behavioral categories adapted from several organizational behavior theories: Interaction Theory,[8] Equity Theory,[1,4] and Multilevel Theory of Resistance to Information Technology.[7] The degrees apart from no usage to high usage and from resistance to support are continuous, not discrete.

Taking a one-dimensional view of acceptance to resistance, as in the previous literature, non-using supporters would be regarded as either IS supporters or non-acceptors ("no usage"), while frequent-using resistors would probably be regarded as accepting the system despite their grudging use or as complete rejectors in case of aggressive resistance. Implementers may not

recognize these two "hidden" groups, viewing them as part of more obvious conventional groups.

Additionally, the figure includes two other groups in the dimension of IS support/resistance. "Other actors" are individuals or groups that can influence and be influenced by IS adoption, even though they are not the intended users.[10] For example, management support can influence users' acceptance or resistance on a particular system, even though managers are not users.[6] Due to the fact they are not intended users, these groups are defined by a single dimension covering "supporting" to "resisting."

Applying the Model

The proposed model can help diagnose intended users' attitudes, as well as other influential stakeholders' attitudes toward IS implementation. The dimension covering acceptance to non-acceptance can be determined by individuals' actual use or by asking about their use intentions. Resistance and support behaviors may be more difficult to trace, especially when passive. Along with alert observation and informal communication, behavioral measurements by means of interviews or surveys represent an effective research tool. Survey questions might include: Do you think using the system will enhance your status or work autonomy? and Do you think using the system will increase your job security? They can be adapted from the literature on resistance toward technology, as in Kappos and Rivard[5] and Markus.[8]

We added two stakeholder reaction types not in the intended user groups but feel they have a stake in the implemented system. Their reactions may be twofold: resist or support adoption and diffusion. For example, patients do not directly use a hospital's electronic record system, but their support or resistance behavior can influence the system's overall success or failure because they can agree or disagree with hospital personnel entering their personal information into electronic records. Such behavior may affect a user's adoption; for example, a doctor may be frustrated with a patient's resistance and look to work around the system.

Case Study

The case we explore here illustrates how to apply the six-group model to governments looking for ways to deliver decent health care to the public in a cost-efficient way. It covers the Dutch government's attempt, begun before 2000, ultimately compulsory in 2012, to implement an EPS for GPs to control prescription costs.

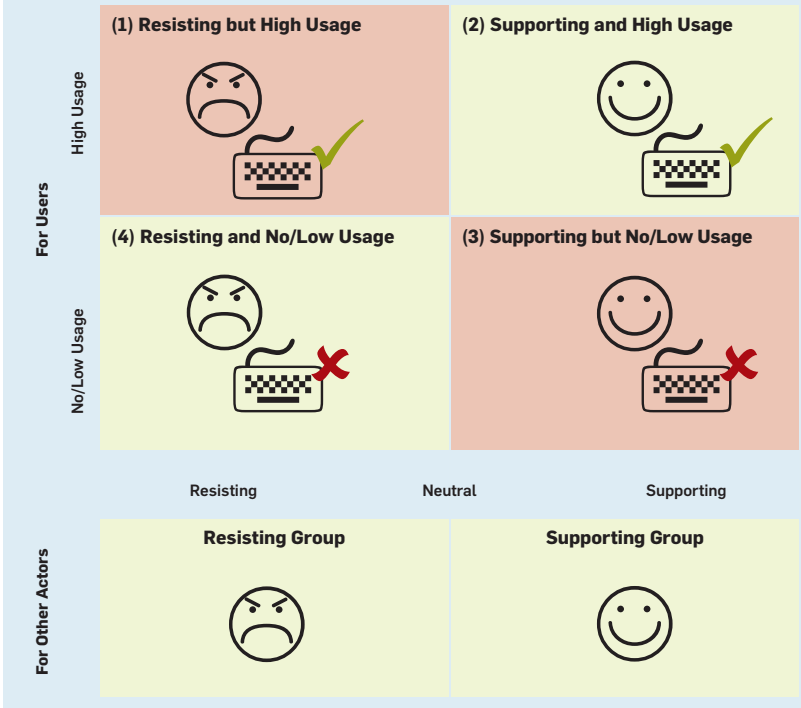
Most Dutch citizens turn to a GP when they need non-urgent medical assistance, with GPs running their practices autonomously as independent businesses. A typical consultation with a patient takes about 10 minutes and steps through four sequences: an introduction, with some informal interaction between GP and patient; a health problem explained by the patient; a diagnosis by the GP in medical terms, possibly coded in the International Classification System of Primary Care, or ICSPC; and a decision by the GP as to proposed treatment, including a drug prescription if necessary.

In the Netherlands, the government and/or health-care insurance providers cover the costs of prescribed drugs. Costs vary up to 40%, depending on quantity, type, and brand. Initial studies had calculated that total annual cost savings would be 20%, or about 136 million Euros, if all Dutch GPs would issue consistent, cost-efficient prescriptions. The government's Ministry of Health Welfare and Sport worked with the country's health insurers and College of General Practitioners and a software company to develop a reliable, user-friendly EPS; the College of General Practitioners negotiated financial benefits for its members in exchange for their cooperation.

Using the EPS to determine appropriate treatment when consulting patients, GPs input their own diagnosis, a list of available drugs, and the patient's personal and medical data. The EPS database includes current drugs, past drugs, drug allergies, drug interactions, and drug costs. The EPS accounts for a patient's specific situation when recommending a treatment, accessing the data when the GP enters the patient's number and diagnosis code from the ICSPC coding system. GPs can then use the EPS to print out and/or email the prescription directly to a pharmacy.

The EPS is designed to advise GPs as to the "best" treatment for a given diagnosis based on the patient's prescription needs, quantity, and most

Figure 1. Six actor groups in adopting IS.



cost-effective brand of medication. The targeted savings were considered feasible if 90% of the country's GPs would adopt the system and follow the recommended treatments in more than 80% of their cases.

To make the system as widely available as possible, the Ministry initiated a large-scale implementation campaign, including instruction videos, booklets, and posters, aiming to educate the country's 8,200 GPs as to how to install and use the system.

Six months following implementation, approximately 50% of surveyed GPs had installed the system, with 50% of this group consulting it at least once a day. However, consulting the system does not necessarily mean a GP would follow its recommendations.

Data was based on our observation of GPs using the system; interviewing 50 of them, along with eight other actors from the implementation team, the Ministry, and health-insurance companies, and reviewing the relevant implementation plans, user manuals, and evaluation studies. We then categorized the interviewed GPs and other relevant stakeholders according to our model (see Figure 2).

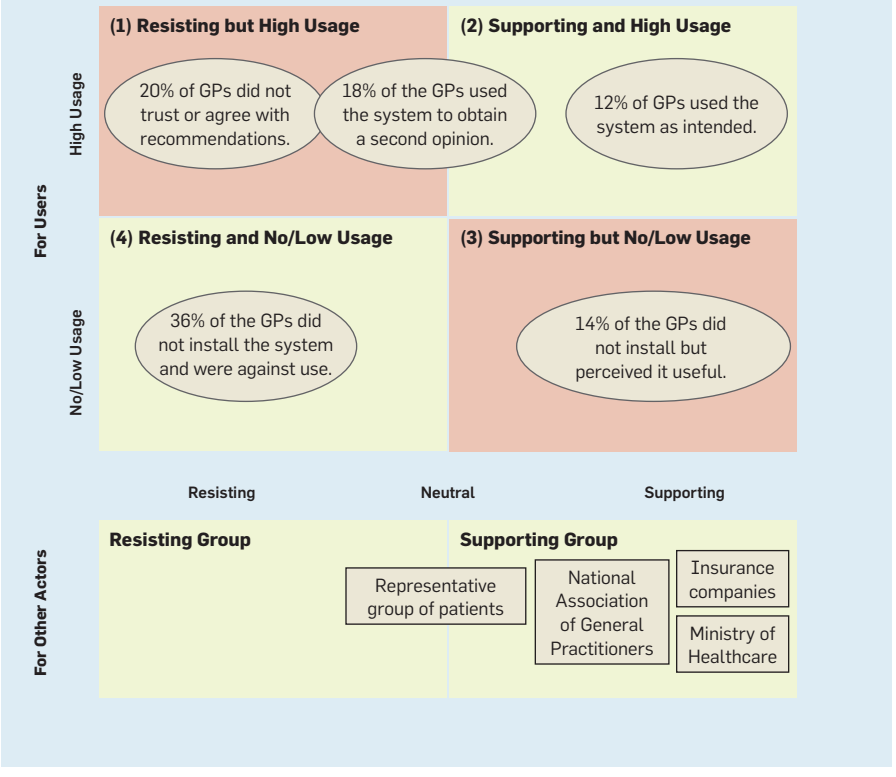
Resisting but high usage. Resisting-but-high-usage GPs perceived problems and disadvantages but still used the system (ambivalent behavior). Some were required to do so by their group-practice managers or colleagues. Typically, they used the system but did not follow its recommendations for several reasons: lack of trust in the recommendations; disagreement with recommended quantity or quality of drugs; or limiting access to obtaining a second opinion after a patient had left the consultation room. The two groups in this quadrant were the 18% of the 50 interviewed GPs who used the system to obtain a second opinion and the 20% who did not trust or agree with the system's recommendations. These groups differed in that GPs from the 18% group used the system to gain confidence determining their patients' prescriptions, though such use did not coincide with the Ministry's publicly stated intentions.

Supporting and high usage. Only 12% of the 50 interviewed GPs said the EPS was easy to use and useful in producing quality output when consulting patients. These supporting-and-high-usage GPs said the system saved them time so they could focus more on the consulting process itself, making consultation more efficient. For example, rather than abruptly conclude a 10-minute consultation with a patient, they could suggest the session was about to end by inserting a special code into the system. These GPs emphasized their prescriptions had become more consistent with those of their colleagues. The result was less time and money spent per patient in group practices and by GPs working part time, as well as by better management of patients' medical data and archives.

Supporting but no/low usage. Among GPs with this ambivalent behavior, 14% openly acknowledged the system's usefulness but said they did not use it due to practical, system-related reasons. Some said they were simply too busy to install it on their computers. Others said they welcomed the idea of using it but lacked the ability or experience to handle the ICSPC coding system that was a prerequisite for use. Yet others deliberately did not have a computer in their consulting rooms to create, they said, a quiet, patient-centered atmosphere but also said they appreciated the idea of a computerized prescription system.

Resisting-and-no/low-usage. Among the 50 interviewed GPs, 36% belonged to the quadrant labeled "resisting and no/low usage." Some viewed the system as complicated, inflexible, and not always available, and consequently felt it would take away from their consultation time with patients—the most medically significant reason indicated for not using the system. Others felt it disrupted their brief contact with patients because they were communicating with the system, not with the patients. In particular, when they had external visits (outside their offices), they did not take a computer with them. Another reason for rejection was financial; to adopt the EPS, they would have had to update or change their computer networks and purchase and implement a new or modified system of patient records. For them, adopting the EPS was simply an additional expense. They also tended to view the EPS as a threat to their social status, feeling politicians and insurers were using it to increase control over their medical practices and weaken their medical autonomy. Other GPs in the same category feared

Figure 2. Result of data analysis.



the system would weaken the perception of therapeutic aura or respect traditionally bestowed on physicians. This could lower the esteem given them by the patients for whom a prescription drug might work as a placebo. Finally, some GPs were concerned about a cultural change in their practices due to innovation, their own initiative, and experimentation to mere compliance with external standards imposed by government administrators.

Resisting and supporting groups. The Ministry and the insurance companies strongly encouraged GPs to adopt and use the system. The College of General Practitioners was supportive though less strongly than other groups. The representative group of patients interviewed shared the GPs' concerns about the system, while other stakeholder groups did not strongly resist EPS diffusion.

This case study illustrates how we could have misunderstood 52% of the GPs interviewed if we had taken a one-dimensional view—acceptance versus resistance. Moreover, two groups within the quadrant labeled "resisting but high usage" showed user reactions are a matter of degree. Implementers should take this into account when developing their strategies.

Promoting Adoption

Here, we assume IS implementers aim to move all users into the supporting-and-high-usage group. In discussing strategies, we focus on the two groups of interviewed GPs with ambivalent behaviors: "resisting but high usage" and "supporting but no/low usage" (see Figure 3):

Resisting but high usage. This ambivalent behavior deserves attention because it is part of an organization's shadow system, with significant influence on overall organizational performance. Though users showing this behavior are considered high usage, it is likely they are forced to use the system, as with 20% of the GPs. In other cases, some people might use a system because they opportunistically experience it as the most convenient option, as 18% of the GPs said they did. These two groups should be treated differently. Users from the 20% group were usually experienced older GPs participating in group practices who felt using the system would go against professional norms valuing personal relationships with patients. They also feared losing autonomy as traditional doctors. Moreover, their concerns were ignored, even as they felt managerial or peer pressure to use the system. To address these concerns, we suggest the following:

Cultural resistance. Potential users can view a system as incompatible with their personal or organizational norms and values. Implementers should thus engage in a dialogue with them to understand those values, explain how the system does not violate them, and cooperate in modifying the system to uphold them; and

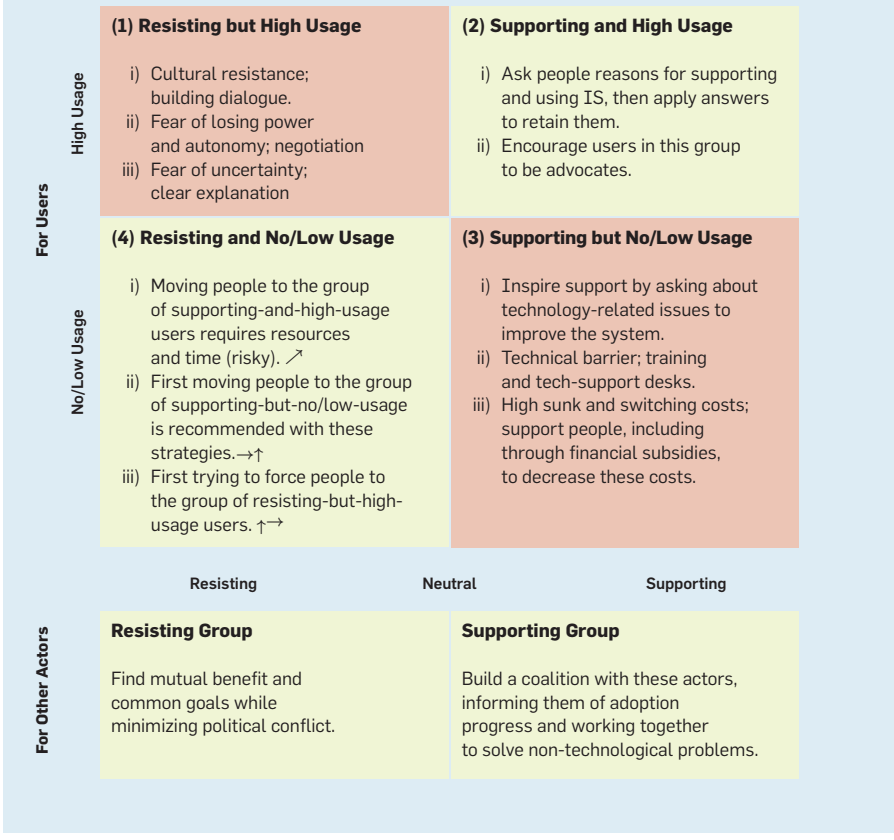
Fear of losing autonomy. Implementers should determine whether concerns about losing autonomy are substantial and legitimate or unfounded. If legitimate, the implementers should negotiate with potential users to achieve a win-win scenario through compensation and other methods. If unfounded, the implementers should do their best to reassure the users they have nothing to fear.

The interviewed GPs from the 18% group were predominantly less-experienced young doctors participating in group practices, feeling underinformed about the consequences of adopting the system, so used it conservatively to obtain a second opinion:

Uncertainty. Anyone can fear and resist a system when ignorant of the consequences of its use. Implementers should explain its purpose and implementation process so the intended users anticipate the changes likely to occur.

Supporting and high usage. Retaining "supporting and high usage" users is as important as attracting users from other groups. Implementers can gather, analyze, and apply the reasons to retain users in this group and attract others from other groups. For example, many young urban GPs from group practices were in this category, believing their use of the EPS made them more professional while increasing medical quality and consistency. Implementers can encourage users in such a group to be advocates by, say, inviting the supporting-and-high-usage GPs to discuss and share their positive experience with the passively resisting-

Figure 3. Strategies for promoting IS adoption, by group.



but-high-usage GPs using the system only to obtain a second opinion.

Supporting but no/low usage. Implementers should beware of viewing this ambivalent group as resisting users just because they do not use the system. Through a mechanism of self-fulfilling prophecy, this group can easily turn into hardened system opponents. Managers and other implementers should explicitly recognize their support and reward it by helping them detect and minimize any related obstacle they may face.

Here, both people- and system-oriented interventions[8] may help:

Materializing support. Implementers can inquire about the technological and practical issues confronting users. The supporting-but-no/low-usage GPs were eager to report their difficulties adopting the system when we interviewed them. Implementers should acknowledge potential users who willingly provide such constructive feedback, telling them how and when the system will be improved to reinforce their promise to use the system.

Technical barriers. People often do not use a system due to lack of skills and/or knowledge. For example, implementers could help the GPs who said they did not know how to use the ICSPC coding system, offering a course through, say, the GPs' own professional association, accrediting it as credit points health professionals must earn annually to maintain their licenses.

High sunk[14] or switching costs.[13] People locked into a particular system find it difficult to switch to any other system. Many of the 50 interviewed GPs reported they could not use the EPS due to incompatibilities with their current computers and networks. Implementers mitigate these barriers by providing extra support to upgrade equipment, including grants and tax deductions.

Resisting and no/low usage. The largest percentage of interviewed GPs (36%) was in this quadrant, mostly experienced doctors practicing independently in small towns who valued their close relationships with their patients, usually knowing them by first name. Moving actors in this quadrant to the desired behavior is especially challenging.

Though implementers try to rationally convince users to adopt supporting-and-high-usage behavior, such an approach may also bring unintended side effects for GPs, in light of their values and limited resources.

Rather than move potential users directly to the "supporting and high" group, implementers should first try to move them to the "supporting but no/low usage" group by following the recommendations outlined earlier in the section on "resisting but high usage." One is to organize open discussions between "resisting and no/low usage" and "supporting and high usage" GPs as to whether the EPS reinforces their values in suggesting more effective drugs for patients. One drawback is the process takes time, even before a GP would start to use the system.

Under some circumstances, implementers can guide potential users toward "resisting but high usage" by forcing them to use the system, as was tried by some managers of GP practices. For independent GPs in small towns, the government could try administrative or financial incentives or even legal means, though the latter would likely provoke even stronger resistance. Therefore, implementers must be prepared to address users' resistance swiftly and sincerely; otherwise, they will fester, eventually hurting overall organizational or professional performance.

Resisting and supporting groups. Much literature, including Kappos and Rivard,[5] Lapointe and Rivard,[7] and Mitchell et al.,[10] has covered how to support other actors (such as through coalitions) and resisting other actors (such as through mutual benefits and common goals). The effort by the government, insurance companies, and College of General Practitioners to win support for the EPS throughout the Netherlands shows IS diffusion does not mean simply providing a technological solution. At times, implementers must mediate political conflicts or acknowledge and address users' emotions concerning the system.

Conclusion

The model introduced here defines ambivalent adoption behavior and explores their occurrence in a real-world context. Moving intended users from low to high use involves system-related support; moving intended users and other actors from resistance to support involves a wider context (such as power shifts, feelings of insecurity, and working environments). These ideas shed light on a gray area previously ignored in the literature on system acceptance and resistance, contributing a new perspective on IS adoption and helping practitioners work with ambivalent groups. We invite researchers to specify the antecedents of ambivalent adoption behavior, as well as the conditions under which intended users might move from quadrant to quadrant. Interaction among the six different groups is another promising area for research. C

References

1. Adams, J.S. Towards an understanding of inequity. *Journal of Abnormal and Social Psychology 67*, 5 (Nov. 1963), 422–436.
2. Ajzen, I. and Fishbein, M. *Understanding Attitudes and Predicting Social Behavior.* Prentice-Hall, Englewood Cliffs, NJ, 1980.
3. Davis, F.D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly 13*, 3 (Sept. 1989), 319–340.
4. Joshi, K. A model of users' perspective on change: The case of information technology implementation. *MIS Quarterly 15*, 2 (June 1991), 229–240.
5. Kappos, A. and Rivard, S. A three-perspective model of culture, information, and their development and use. *MIS Quarterly 32*, 3 (Sept. 2008), 601–634.
6. Kim, H.W. and Kankanhalli, A. Investigating user resistance to information systems implementation: A status quo bias perspective. *MIS Quarterly 33*, 3 (Sept. 2009), 567–582.
7. Lapointe, L. and Rivard, S. A multilevel model of resistance to information technology implementation. *MIS Quarterly 29*, 3 (Sept. 2005), 461–491.
8. Markus, M.L. Power, politics, and MIS implementation. *Commun. ACM 26*, 6 (June 1983), 430–444.
9. Markus, M.L. and Mao, J. Participation in development and implementation: Updating a tired, old concept for today's IS contexts. *Journal of the AIS 5*, 11–12 (Nov. 2004), 514–544.
10. Mitchell, R.K., Agle, B.R., and Wood, D.J. Toward a theory of stakeholder identification and salience: Defining the principle of who or what really counts. *Academy of Management Review 22*, 4 (Oct. 1997), 853–886.
11. Nagy, D., Yassin, A.M., and Bhattacherjee, A. Organizational adoption of open source software: Barriers and remedies. *Commun. ACM 53*, 3 (Mar. 2010), 148–151.
12. Nah, F.F., Tan, X., and Teh, S.H. An empirical investigation on end users' acceptance of enterprise systems. *Information Resources Management Journal 17*, 3 (July–Sept. 2004), 32–53.
13. Seo, D., Ranganathan, C., and Babad, Y. Two-level model of customer retention in the U.S. mobile telecommunications service market. *Telecommunications Policy 32*, 3–4 (Apr. 2008), 182–196.
14. Sutton, J. *Sunk Costs and Market Structure.* MIT Press, Cambridge, MA, 1991.
15. Venkatesh, V. and Davis, F.D. A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management Science 46*, 2 (Feb. 2000), 186–204.

DongBack Seo (d.seo@rug.nl) is an assistant professor of management information systems in the Faculty of Economics and Business at the University of Groningen, the Netherlands.

Albert Boonstra (albert.boonstra@rug.nl) is a professor of information management at the Faculty of Economics and Business at the University of Groningen, the Netherlands.

Marjolein van Offenbeek (m.a.g.van.offenbeek@rug.nl) is an assistant professor of organizational behavior and technological change in the Faculty of Economics and Business at the University of Groningen, the Netherlands.

DOI:10.1145/2018396.2018415

HPF pioneered a high-level approach to parallel programming but failed to win over a broad user community.

BY KEN KENNEDY, CHARLES KOELBEL, AND HANS ZIMA

The Rise and Fall of High Performance Fortran

PARALLELISM—THE EXECUTION of multiple tasks at the same time—is fundamental in computer design. Even very early computer systems employed low-level parallelism (such as overlapping input-output with computing). Later designs moved parallelism to higher levels, including vector processors in the 1970s and shared-memory and distributed-memory parallel supercomputers in the 1980s. Software support for these machines never fully reached the standard of commercial software development, due mainly to the specialized market for high-performance computing and the difficulty related to users' extreme focus on target-code performance. Here, we trace the history of High Performance Fortran (HPF), a high-level data-parallel language standardized by a group of industry and academic leaders, 1991–1993. It was initially well received by the community, generating a great deal of interest in the language and its implementation, but never succeeded in building a strong user base. We concentrate on the technical and social process of designing HPF, as well as the factors that contributed to its successes and its shortcomings.

HPF was one of many efforts to develop parallel programming techniques active since the late 1980s. Here, we do not treat these efforts comprehensively or review all work prior to HPF; more on these topics can be found in the conference version of this article[15] and in the History of Programming Languages conferences (http://www.acm.org/sigplan/hopl). Rather, we provide background on the parallel computers and programming systems available when HPF was developed, then turn to the process of defining HPF, the features of the language itself, and its successes and failures. We conclude with a discussion of lessons to be learned by developers and users of other languages.

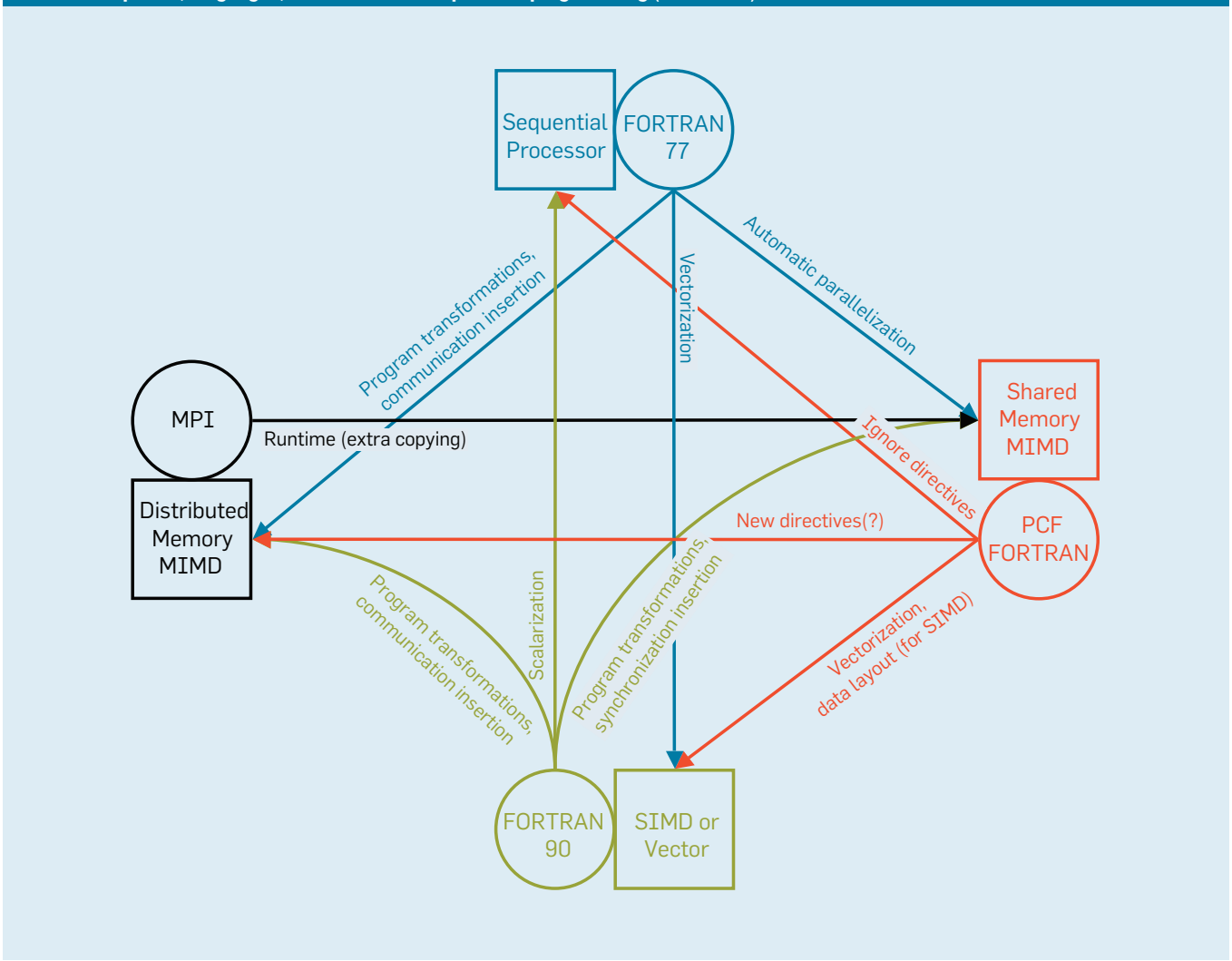
Parallel Computers and Languages

The milieu in which HPF was created (see the figure here) guides much of our discussion, with rectangles representing classes of computers and attached circles representing the most common programming languages for

» key insights

- **Programming high-performance parallel computing systems has been dominated by a low-level approach requiring explicit management of communication; HPF elevated parallel programming to a higher level of abstraction, enhancing portability, programmer productivity, and dependability.**
- **Many ideas pioneered by HPF (such as high-level data distribution directives and parallel loop constructs) have significantly influenced modern parallel languages, including Chapel, X-10, and Fortress.**
- **HPF's expressive power led to early success, but program performance could not always match that of MPI-based programs, resulting in failure to win over a broad user community.**

Parallel computers, languages, and the search for portable programming (circa 1990).



each class. A line from a language L to an architecture A represents the compiler/runtime or transformation technology needed to implement L on A. For completeness, we include sequential computers and FORTRAN 77.

The data-parallel computation model is characterized by the property that sequences of operations or statements can be performed in parallel on each element of a collection of data. This model was in part motivated by the work on the NESL language.[4] Two common implementations were vector processors (featuring highly pipelined CPUs operating on vector registers) and single instruction multiple data (SIMD) computers (featuring thousands of simple processors operating under a single control unit). Due to this fine-grain control, the most natural languages for these machines provided array arithmetic and array functions mirroring the vector and matrix operations in many scientific codes.

Following years of development, vectorizing compilers were able to identify and exploit this level of parallelism in FORTRAN 77 programs. The key enabling techniques were dependence analysis, which identified conflicting operations on the same data that would prevent parallel execution, and compiler transformations, which reorganized operations in loops to enhance the opportunities for vector operation. The result was that many users of vector machines continued programming in a sequential language.

However, due to the need to identify and exploit much higher levels of parallelism, purely automatic parallelization was not as successful on SIMD machines. Programmers on these architectures wrote in explicitly data-parallel languages (such as NESL and Connection Machine Fortran, or CMF[22]). Later, Fortran 901 represented an important step toward data-parallel languages by standardizing array operations, array assignments, and intrinsic data-parallel functions (such as reductions like `SUM`). HPF was later based on this programming model.

Multiple instruction multiple data (MIMD) computers allowed more general and larger-grain parallel execution by providing each processor its own instruction stream and the ability to operate asynchronously to all other processors.

MIMD computers could be classified as either shared-memory or distributed-memory architectures. In shared-memory MIMD machines all processors were connected to a single shared main memory, making it simple to access common data and communicate results between processors. Languages for these machines featured a global view of data and mul-

tiple ways to create parallel activities (such as parallel loops in which iterations could execute concurrently) and synchronization operations for enforcing order between activities on different processors. The Parallel Computing Forum promulgated a standard set of FORTRAN extensions for this programming model[18] to allow easy programming while enabling access to the performance advantages of shared memory over other machines.

However, two sets of problems arose with these architectures: First, the inability to hide latency with conventional processors at large scale and the hardware overhead of implementing cache coherence limiting their scalability; and second, the possibility of data races, so if two parallel threads access the same memory location, with at least one of them writing to that location, then a lack of proper synchronization could lead to different results for different parallel schedules.

These problems led to development of distributed-memory MIMD machines in which processors were interconnected via networks, with each processor having its own local memory. This organization allowed the machines to scale to around 1,000 processors by the early 1990s. The price of this architectural simplicity was software complexity; when two processors would have to share data, they would have to exchange messages, an expensive operation. Reducing this cost relied on correctly placing the data to minimize the required communication, and placing the message-passing calls in the most appropriate program locations. By the time the HPF effort had begun in the early 1990s, message-passing libraries (such as PVM and PARMACS) were already being used for programming MIMD systems in connection with standard sequential languages. A standardization activity for message passing began soon thereafter, patterned after and in close connection with the HPF effort.[11] The resulting Message Passing Interface (MPI)[20] library provided efficient explicit control of locality and communication. Despite the greater complexity of this programming paradigm, it quickly became popular due to its ability to exploit the performance of distributed-memory computers.

The HPF goal was a common programming model that could execute efficiently on all classes of parallel machines.

The HPF goal was a common programming model that could execute efficiently on all classes of parallel machines. Developers had to answer whether the advantages of shared memory, even a single thread of control, can be simulated on a distributed-memory machine; also how parallelism can be made to scale to hundreds or thousands of processors. Such issues were addressed through data-parallel languages in which the large data structures of applications are part of a global name space that can be laid out across the memories of a distributed-memory machine, or a "data distribution." The data elements mapped to the local memory of a processor in this way are said to be "owned" by the processor. Program execution is modeled by a single thread of control, but the components of distributed data structures can be operated on in parallel. The data distribution controls how work is to be allocated among the processors. Compilation techniques (such as those discussed later) allowed MIMD architectures to relax the lockstep semantics of the SIMD model to improve efficiency.

The HPF design was based largely on experience with the language designs and implementations of the early data-parallel languages. Here, we focus on the data-parallel approach embodied in HPF. To be sure, it was not universally embraced by either the compiler research community or the application community. Other programming paradigms, including functional and dataflow languages, also had substantial intellectual merit, passionate user communities, and/or both.

HPF Standardization Process

At the Supercomputing conference in 1991 in Albuquerque, NM, Ken Kennedy of Rice University and Geoffrey Fox of Indiana University met with a number of commercial vendors to discuss the possibility of standardizing the data-parallel versions of Fortran. These vendors included Thinking Machines (then producing SIMD machines), Intel and IBM (then producing distributed-memory MIMD machines), and Digital Equipment Corp. (then interested in producing a cross-platform Fortran system for both SIMD and MIMD machines).

Kennedy and Fox organized a birds-

of-a-feather session with academic and industrial representatives who agreed to explore a more formal process through a group that would come to be known as the High Performance Fortran Forum (HPFF), with Kennedy serving as chair and Charles Koelbel as executive director. With support from the Center for Parallel Computation Research (CRPC) at Rice University, a meeting with nearly 100 participants organized in Houston in January 1992 concluded with a business session in which more than 20 companies committed to a process for drafting the new standard.

They agreed the process should produce a result in approximately one year. It turned out this tight schedule would affect the language, leading to adoption of the rule that HPF would include only features that had been demonstrated in at least one language and compiler, including research compilers. This limited some of the features that could be considered, particularly in the realm of advanced data distributions.

The 30 to 40 active HPFF participants then met for two days every six weeks or so, mostly in a hotel in Dallas. Besides Kennedy and Koelbel, the participants serving as editors of the standard document included Marina Chen, then at Yale, Bob Knighten, then at Intel, David Loveman, then at DEC, Rob Schreiber, then at NASA, Marc Snir, then at IBM, Guy Steele, then at Thinking Machines, Joel Williamson, then at Convex, and Mary Zosel, then at Lawrence Livermore National Laboratory. Attendees represented five government labs, 12 universities, and 18 companies (vendors and users); they also hailed from at least five countries. HPFF was a consensus-building process, not a single-organization project.

It dealt with numerous difficult technical and political issues, many due to the tension between the need for high-level language functionality supporting a broad range of applications and the need to generate efficient target code. They were compounded by limited experience with the compilation of data-parallel languages. The research compilers were mostly academic prototypes that had been applied to few applications of any size. On the other hand, the industrial-strength language CMF was relatively new and lacked some advanced features of the research languages. Many decisions thus had to be made without a full understanding of their effect on compiler complexity.

The result was a new language finalized in early 1993[12] and presented later that year at the Supercomputing conference in Portland, OR. An additional set of HPFF meetings was held in 1994, aiming to address corrections and clarifications of the existing standard and consider new features for the support of additional applications. Ken Kennedy was again chair, while Mary Zosel was now executive director. The attendees were mostly the same as in the 1992–1993 meetings, with notable additions Ian Foster of Argonne National Laboratory and Joel Saltz, then at the University of Maryland, each leading new subgroups. The corrections were included in the HPF 1.1 standard, a slight revision of the 1993 document presented at the 1994 Supercomputing conference in Washington, D.C. Some new features and clarifications discussed but not included in HPF 1.1 were collected in the *HPF Journal of Development* and later served as a starting point for the HPF 2.0 standardization effort, 1995–1996.

This effort would incorporate new features, most notably the ability to perform accumulations in the `INDEPENDENT` loop construct, advanced data distributions, and the `ON` clause, which provided control over computation mapping by identifying the processor to execute each iteration of a parallel loop. HPF 2.0 also defined "approved extensions" considered too complex to be required initially in all compilers. HPFF had intended that vendors would implement all "core language" features at first, then prioritize extensions based on customer demand. Not surprisingly, the result was confusion, as the feature sets offered by different vendors diverged.

HPF Language

The goals established for HPF were fairly straightforward: provide a high-level, portable programming model for scalable computer systems based (primarily) on data-parallel operations in a (conceptually) shared memory and produce code with performance comparable to the best hand-coded native language code on a given machine. To achieve them, HPF 1.0 defined a language with several novel characteristics. For simplicity, we do not discuss the enhancements in HPF 1.1 or HPF 2.0, though they were in much the same vein.

First, the language was based on Fortran 90,[1] with extensions defined as a set of directives in the form of structured comments. These directives could be interpreted by HPF compilers as advice concerning how to produce a parallel program. On a scalar machine, an HPF program could be executed without change simply by ignoring the directives, assuming the machine had sufficient memory. This device permitted the HPF parallelism extensions to be separated cleanly from the underlying Fortran 90 program; the same program would work on both sequential and parallel systems. Sequential-to-parallel portability is a huge advantage in debugging a code.

The principal additions to the language were a set of distribution directives. Sets of arrays could be aligned with one another, then distributed across the processors through built-in distributions. These directives could be used to assign the individual rows or columns of an array to processors in contiguous blocks or in round-robin fashion. This feature is illustrated by showing the application of directives to two simple Fortran arrays

```
REAL A(1000,1000), B(1000,1000)
```

Now suppose we want to split the first dimension (and with it, the computation over that dimension) across the processors of a parallel machine. Moreover, suppose because corresponding elements of `A` and `B` are often accessed together, they should always have the same distribution. Both effects can be accomplished through the directives

```
!HPF$ DISTRIBUTE A(BLOCK,*)
!HPF$ ALIGN B(I,J) WITH A(I,J)
```

HPF also provides the `CYCLIC` distribution, in which elements are assigned to processors in round-robin fashion, and `CYCLIC(K)`, in which blocks of `K` elements are assigned round-robin to processors. Generally speaking, `BLOCK` is the preferred distribution for computations with nearest-neighbor elementwise communication, whereas the `CYCLIC` variants

allow finer load balancing of some computations.

Data distribution of subroutine arguments was particularly complex. Formal subroutine arguments could be associated with ALIGN and DISTRIBUTE directives. If they fully specified a data distribution, then the corresponding actual arguments would be redistributed when the call was made and redistributed back to their original distribution on return. If the caller and callee distributions matched, it was expected that the compiler or runtime system would forego the copying needed in the redistribution. HPF also defined a system of "inherited" distributions in which the distribution of the formal arguments would be identical to the actual arguments. This declaration required an explicit subroutine interface (such as a Fortran 90 INTERFACE block). In this case, copying could be avoided, but code generation for the subroutine would have to handle all possible incoming distributions. The complexity was so great that to our knowledge no compiler fully implemented it.

In addition to distribution directives, HPF provides directives to assist identification of parallelism, illustrated with a simple smoothing operation on the arrays described earlier

```
DO J = 2, N
  DO I = 2, N
     A(I,J)=(A(I,J+1)+2*A(I,J)+A(I
     ,J-1))*0.25 &
&          +(B(I+1,J)+2*B(I,J)+B(I-
           1,J))*0.25
  ENDDO
ENDDO
```

Using Fortran 90 array notation produces this code

```
DO J = 2, N
   A(2:N,J) = &
&    (A(2:N,J+1)+2*A(2:N,J)+A(I
     ,J-1))*0.25 &
&    +(B(3:N+1,J)+2*B(2:N,J)+B(1:N-
     1,J))*0.25
ENDDO
```

HPF also provided a FORALL statement, taken from CMF, as an alternative means of expressing array assignment. The inner DO loop in our relaxation example could be written as

```
FORALL ( I=2:N ) &
&  A(I,J)=(A(I,J+1)+2*A(I,J)+
   A(I,J-1))*0.25 &
&        +(B(I+1,J)+2*B(I,J)+B(I-
         1,J))*0.25
```

The explicit indexing allowed FORALL to conveniently express a wider range of array shapes and computations than the standard array assignment.

HPF included the ability to specify that the iterations of a loop should be executed in parallel using the INDEPENDENT directive. While in this inner loop the compiler could easily derive this property, and subscripted subscripts would in general require runtime analysis without the explicit assertion provided by the INDEPENDENT directive. The programmer often has application-specific knowledge that would allow such loops to be executed in parallel, like this

```
DO J = 2, N
!HPF$ INDEPENDENT
  DO I = 2, N
   A(INDX(I),INDY(J))= ...
  ENDDO
ENDDO
```

HPF was also one of the first language specifications to include an associated library, the HPF Library, as a part of its definition, adding power to the language by providing parallel operations on global arrays (such as sum reduction, gather/scatter, and partial prefix operations).

Finally, HPF included features designed to improve compatibility and facilitate interoperation with other programming languages and models. In particular, the EXTRINSIC interface made it possible to invoke subprograms written in other languages (such as scalar Fortran and C). Of particular importance was the ability to call subroutines written in MPI in a way that made it possible to recode HPF subprograms for more efficiency.

HPF Compilation Technology

The concepts underlying the HPF language design and related compilation technology were pioneered by the early data-parallel languages preceding HPF, particularly CMF,[22] Fortran D,[13] and Vienna Fortran.[6] The key approach common to them was adoption of a high-level notation for specifying data distribution, thus relegating the task of explicit generation of communication to the compiler and runtime system. Fortran D and Vienna Fortran initially adopted the owner-computes rule for compilation to distributed-memory MIMD machines. Accordingly, the compiler generates code for each statement in such a way that all computations are performed on the processors owning the computation output. Later, this strategy was modified to perform computations on any processors that would minimize communication. The rule is complemented by a general execution model in which each processor scheduled to perform a computation performs the following three steps: generate messages to send data it owns to all other processors that need them; await arrival of messages from other processors containing required nonlocal data; and complete the computation using its local data and non-local data communicated from other processors. The point-to-point messages generated through this paradigm are all the synchronization required for a distributed-memory MIMD implementation. This idea was so successful that, when Thinking Machines introduced its MIMD CM-5 computer in 1993, it retained the data-parallel CM Fortran language. Similar techniques could generate the synchronization needed for shared-memory MIMD computers.

Here, we outline the major phases in an HPF compiler that performs a source-to-source transformation:[3,14]

Following the compiler front end, the data-distribution phase analyzes HPF mapping directives to determine ownership of all data objects. Reaching distribution analysis serves to associate all occurrences of arrays with distribution information, including dummy arrays. If ownership cannot be resolved statically, code is generated to determine this information at runtime.

The work-distribution phase generates an explicitly parallel single-program-multiple-data (SPMD) program by determining the distribution of work and inserting communication. In it, the owner-computes paradigm is enforced, and for potential nonlocal accesses, communication primitives are inserted that transfer nonlocal data into private variables.

A key goal of the optimization phase is reduction of communication overhead via a range of techniques, including execution of communication in parallel to computation, elimination of redundant communication, and exploitation of collective communication primitives whenever possible. Overlap analysis detects simple communication patterns (such as stencils), using this information to improve local data management, as well as the organization of communication. Gupta et al.[10] presented a general framework for optimizing communication in data-parallel programs. In the final code-generation phase, the optimized parallel program is transformed into a Fortran program with explicit message passing.

The inspector-executor paradigm[17] is an important method for runtime optimization of parallel loops not amenable to static analysis due to irregular array accesses (such as through subscripted subscripts).

At its peak, 17 vendors offered HPF products and more than 35 major applications were written in HPF, at least one with more than 100,000 lines of code.

Experience with the Language

The initial response to HPF can be characterized as cautious enthusiasm. A large part of the high-performance user community was hopeful that the high-level abstractions provided by the language would make parallel programs portable and efficient without requiring explicit control of message passing. On the other hand, vendors hoped HPF would expand the market for scalable parallel computing. Several major vendors, including DEC, IBM, and Thinking Machines, initiated independent compiler efforts; others offered OEM versions of compilers produced by independent software companies (such as Applied Parallel Research and the Portland Group, Inc.). At its peak, 17 vendors offered HPF products and more than 35 major applications were written in HPF, at least one with more than 100,000 lines of code.

Much HPF experience was reported at meetings of the HPF Users Group in Santa Fe, NM (1997), Porto, Portugal (1998), and Tokyo (2000).[16] The Tokyo meeting was notable for demonstrating the strong interest in HPF in Japan, later reemphasized by the Earth Simulator[19] featuring a high-functionality HPF implementation supporting the HPF/JA extensions. The IMPACT-3D fluid simulation for fusion science on the Earth Simulator achieved 40% of its peak speed and was awarded a Gordon Bell Prize at the Supercomputing conference in 2002 in Baltimore.

As the language was applied to a more diverse set of application programs it became clear that in many cases its expressive power allowed the formulation of scientific codes in a clearer, shorter, less error-prone way than was possible based on explicit message passing. HPF did best on simple, regular problems (such as dense linear algebra and partial differential equations on regular meshes). However, for some data-parallel applications it was difficult to achieve the all-important goal of high target-code performance. As a result, frustrated users, particularly in the U.S., switched to MPI. This migration significantly reduced demand for HPF, leading compiler vendors to reduce, even abandon, their development efforts. The end result was HPF never achieved a sufficient level of acceptance among leading-edge users of high-performance parallel computing systems.

Here, we explore the main reasons for this development:

Missing language features. To achieve high performance on a variety of applications and algorithms, a parallel programming model must support a range of different kinds of data distributions. The original HPF specification included three main classes of such distributions—`BLOCK`, `CYCLIC`, and `CYCLIC(K)`—motivated largely by the requirements of regular algorithms operating on dense matrices and well adapted to the needs of linear algebra. However, many relevant applications need more general distributions that deal efficiently with dynamic and irregular data structures. Examples include multiblock and multigrid codes, finite element codes (such as crash simulations operating on dynamically changing unstructured grids), spectral codes (such as weather forecasting), and distributed sparse matrix codes. Algorithmic strategies in these codes were difficult or even impossible to express within HPF without sacrificing too much performance. Though this problem was addressed to a certain degree in HPF 2.0 through basic support for irregular data distributions, the damage was done.

Another language issue with HPF was limited support for task parallelism. Users wanted more powerful strategies beyond the parallel loop feature offered in HPF. This was corrected in HPF 2.0 but too late for a revival of HPF. Including features like the OpenMP set of directives could have helped.

Immature compiler technology. HPF was defined on top of Fortran 90, the first major upgrade to the Fortran standard since 1977. Among the new features of Fortran 90 were explicit interfaces requiring array descriptors, recursion, dynamic storage allocation, and pointer-based data structures. Building a Fortran 90 compiler meant a substantial implementation effort compared with FORTRAN 77—a huge obstacle on the way to HPF.

Moreover, the features in HPF, including the HPF library, demanded new compilation strategies that in 1993 at the time of the release of HPF 1.0 had been implemented only in research compilers and the CM Fortran compiler. Proper compilation of HPF requires extensive global analysis of distributions, partitioning of computation, generation of communication, and optimizations (such as overlapping communication and computation).

Finally, efficient implementation of HPF programs required special attention to locality on individual processors. Since many of the processors used in distributed-memory systems had complex cache hierarchies, advanced transformation strategies (such as loop tiling) were becoming essential to generate efficient code. At the time of the release of HPF 1.0, these techniques were beginning to be understood by compiler developers, but most commercial compilers had not yet incorporated them. As a result the first compilers were immature, providing disappointing performance on many codes.

Barriers to portable performance. A key goal of HPF was enabling a single version of a parallel program to achieve a significant fraction of the possible performance on a variety of architectures but was difficult to achieve for two main reasons:

Different optimizations. Different vendors focused on different optimizations in their implementations, causing a single HPF application to deliver dramatically different performance on machines from different vendors. In turn, programmers were forced to recode their applications repeatedly to take advantage of the strengths (and avoid the weaknesses) of each vendor's implementation, thwarting the original goal of portability.

The end result was HPF never achieved a sufficient level of acceptance among leading-edge users of high-performance parallel systems.

No reference implementation. Second, the HPF Library could have been used to address some of the usability and performance problems described earlier, but there was no open-source reference implementation for the library, so each compiler project had to implement its own version. Due to the number and complexity of library components, this was a significant burden. The end result was that the implementations were inconsistent and often exhibited poor performance; users were again forced to code differently for different target machines.

Difficulty of performance tuning. Every HPF compiler we know of translated the HPF source to Fortran plus MPI. In this process, a large number of transformations were carried out, making the relationship between the original source program and the corresponding target program less than obvious to the programmer, making it very difficult to identify and correct performance problems. Though research groups were eventually able to apply MPI-based performance tools to identify bottlenecks in HPF programs, the tuning problem was still not addressed. That is, the user might well understand what was causing the performance problem but have no idea how to change the HPF source to address it.

Significant Influence

HPF also had significant influence on development of high-level parallel languages. In 2007, the CiteSeer database listed 827 citations for the language definition,[12] making it the 21st most-cited document in computer science at the time. In addition, more than 1,500 publications in CiteSeer refer to "High Performance Fortran," many including approaches to implementing or improving the language, reflecting a great deal of intellectual activity within the academic community.

Fortran and its variants. *Fortran 95.* While the meetings leading to HPF 1.1 were under way, the X3J3 committee of

the American National Standards Institute (ANSI, http://ansi.org/) developed the Fortran 95 standard. When formally adopted in 1996, it included the HPF FORALL and PURE features nearly verbatim. The HPF contributions to Fortran have been retained in all Fortran standards since then.

HPF/JA. In 1999, the Japan Association for High Performance Fortran, a consortium of Japanese companies, including Fujitsu, Hitachi, and NEC, released HPF/JA,[19] with features found in previous programming languages on parallel-vector machines from Hitachi and NEC. An important source contributing to HPF/JA was the HPF+ language[2] implemented in a European project led by the Vienna group, with NEC as a project partner. HPF+, based on an analysis of advanced industrial codes, provided a REUSE clause for independent loops asserting reusability of the communication schedule computed during the first iteration of the loop. In the same context, its HALO construct (renamed the REFLECT directive in HPF/JA) allowed the functional specification of non-local data accesses in processors and programmer control of the copying of such data to region boundaries. The LOCAL directive could be used to specify that communication was not needed for data access in some situations, a fact a compiler might be unable to recognize. These and other features allowed for better control over locality in HPF/JA programs. HPF/JA was later implemented on the Earth Simulator.[19]

OpenMP. OpenMP[8] was proposed at the end of 1997 as an extension of Fortran, C, and C++, providing a set of directives that support a shared-memory programming interface, extending earlier work by the Parallel Computing Forum as part of the X3H5 standardization committee[18] and SGI directives for shared-memory programming.

As in the Parallel Computing Forum directives, OpenMP provides a means to generate threads for the processors of a shared-memory machine and control access to shared data in an efficient manner. However, OpenMP does not provide features to control locality like HPF's distribution directives. Attempts were made to address this shortcoming by integrating OpenMP with language features that allow locality-aware programming. However, the current OpenMP standard does not reflect any of them.

HPCS languages. Key HPF ideas are still finding their way into newer programming languages, particularly in DARPA's High Productivity Computing Systems (HPCS) program, which aims to ease the burden of programming leading-edge systems based on innovative hardware and software architectures. From HPCS came three new language proposals: Chapel,[5] Fortress,[21] and X10.[7] All are object-oriented languages supporting a range of features for programmability, parallelism, and safety, along with a global name space, explicit multithreading, and explicit mechanisms for dealing with data parallelism and locality. Each execution of a program is bound to a set of virtual locality units mapped by the operating system to physical entities (such as computational nodes). This gives the programmer a way to distribute data collections across locality units, align different collections of data, and establish affinity between computational threads and the data on which they operate. This approach represents a generalization of key elements in HPF, while the data-parallel features in the languages address many of the shortcomings described earlier. In particular, they offer a wider range of constructs for expressing parallelism and synchronization and do not rely nearly as much on advanced compiler optimizations.

The evolution of parallel loop constructs also illustrates the transition from HPF to HPCS languages. For example, the HPF INDEPENDENT construct asserts that the loop does not contain loop-carried dependences, thus excluding data races and allowing correct parallel execution. Chapel distinguishes between a sequential for loop and a parallel forall loop that iterates over the elements of an index domain without restriction on loop-carried dependences. Thus, programmers are responsible for avoiding dependences that lead to data races. The Fortress for-loop is parallel by default, and if a loop iterates over a distributed dimension of an array the iterations are grouped onto processors according to the distributions. A special "sequential" distribution can be used to serialize a for-loop. X10 distinguishes two kinds of parallel loops: the foreach loop, which is restricted to a single locality unit, and the ateach loop, which allows iteration over multiple locality units. Again, the programmer must use the construct correctly to avoid data races.

Parallel scripting languages. Though the Fortran and C communities were willing to tolerate the difficulty of writing MPI code for scalable parallel machines, it seems unlikely that the large group of programmers of high-level scripting languages (such as Matlab, Python, and R) would be willing to do the same. Simplicity is part of the reason for the popularity of these languages.

Nevertheless, there is substantial interest in being able to write parallel code in the languages. As a result, a number of research and commercial projects have explored strategies for parallel programming, particularly in Matlab. Most replace the standard Matlab array representation with a global distributed array and provide replacements for all standard operators performing distributed operations on these arrays. Though it is in the spirit of HPF, the overhead of producing operation and communication libraries by hand limits the number of different distributions such systems are able to support. Many current implementations are therefore restricted to a variant of the general block distributions in HPF 2.0.

The goal of these efforts is to provide the scripting-language community a simple way to obtain scalable parallelism with minimal change to their programs. If they succeed, they will not only vindicate the HPF vision but dramatically increase the community of application developers for scalable parallel machines.

Lessons Learned

HPF aimed to introduce a high-level programming notation to enhance portability and programmer productivity while offering target code performance meeting the expectations of a highly sophisticated user community. HPF did succeed in the first of these goals but failed to achieve performance competitive with MPI-based programs for many important applications. This shortfall became apparent, particularly after MPICH (http://www.mcs.anl.gov/

research/projects/mpich2/), a portable, efficient reference implementation of MPI, became available.

The dominant technical reasons for failing to achieve performance competitive with MPI-based programs involved language design, compiler technology, and tool support. First, as discussed earlier, the data distributions provided by HPF could not adequately support large classes of important applications. No particular set of built-in distributions can satisfy all these requirements; rather, what is needed is a general mechanism for generating programmer-defined distributions via an abstraction that can be seen in analogy to the control abstraction created by function definitions. Such a capability could be used to extend a library of standard distributions, with distribution libraries that exploit the specific data structures and access patterns reflecting properties of an application domain or architecture. An approach along these lines was analyzed by Diaconescu and Zima[9] in the context of Chapel language development.

Second, compiler technology for generating high-performance target codes was not mature enough at the beginning of the HPF effort. This relates to the absence of industrial-strength implementations of Fortran 90 at that time but also to not understanding how to generate efficient target code for many HPF language features. Despite considerable research addressing these features, there was also a significant lack of practical experience. One important aspect of it was the HPF library, originally envisioned as a pillar of support for efficient HPF coding but never completely implemented. The result was the programmer could not rely on the HPF Library if efficiency was a concern, thus reducing the language's overall usability.

A third set of problems involved tools for performance tuning. Collaboration between compiler and tool developers makes it possible to map performance information back to the HPF source. However, a programmer had only limited ways to exercise fine-grain control over the code generated once the source of performance bottlenecks was identified, other than using the `EXTRINSIC` interface to drop into MPI. The HPF/JA extensions ameliorated this problem by providing more control over locality. However, additional language features were needed to override compiler actions if necessary.

Finally, some general circumstances have also worked against HPF. First, the HPC market is fairly small; as a consequence, it has always been difficult to obtain funding for the development of industrial-strength compilers and runtime systems. Second, Fortran use seems to be shrinking, so HPF may have suffered due to its being bound to Fortran. And third, developers of large-scale applications make investments that are supposed to be useful for decades; unlike MPI and its broad user and vendor support, long-term HPF viability were indeed less assured from the very beginning.

Conclusion

HPF has succeeded over the past 20 years in the sense it had a significant influence on language and compiler technology for parallel architectures. It included ideas and technologies that have become part of the next generation of high-performance computing languages. These include a global view of data and control, high-level specification of data distributions, interoperability with other language models, and an extensive library of primitive operations for parallel computing. Many of the original implementation impediments were resolved as a result of advanced compiler and runtime technology developed since HPF inception. Nevertheless, for both technical and nontechnical reasons, the language failed to be adopted by a broad user community. Perhaps the time is right today for the high-performance community to again embrace new parallel-programming models, like the one supported by HPF.

Acknowledgments

We thank the referees for their thoughtful, detailed, constructive comments that made a significant contribution to improving the final version of this article. C

References

1. American National Standards Institute X3J3/S8.115. Fortran 90, June 1990; http://www.ansi.org
2. Benkner, S., Lonsdale, G., and Zima, H.P. The HPF+ project: Supporting HPF for advanced industrial applications. In *Proceedings EuroPar Parallel Processing, Vol. 1685 of Lecture Notes in Computer Science*. Springer-Verlag, 1999.
3. Benkner, S. and Zima, H. Compiling High Performance Fortran for distributed-memory architectures. *Parallel Computing 25* (1999), 1785–1825.
4. Blelloch, G.E. *NESL: A Nested Data-Parallel Language, Version 3.1.* Technical Report CMU-CS-95-170. School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Sept. 1995.
5. Chamberlain, B.L., Callahan, D., and Zima, H.P. Parallel programmability and the Chapel language. Special issue on high-productivity languages and models. *International Journal of HPC Applications 21*, 3 (2007), 291–312.
6. Chapman, B., Mehrotra, P., and Zima, H. Programming in Vienna Fortran. *Scientific Programming 1*, 1 (Fall 1992), 31–50.
7. Charles, P., Grothoff, C., Saraswat, V., Donawa, C., Kielstra, A., Ebcioglu, K., von Praun, C., and Sarkar, V. X10: An object-oriented approach to non-uniform cluster computing. In *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications*. ACM Press, New York, 2005, 519–538.
8. Dagum, L. and Menon, R. OpenMP: An industry-standard API for shared-memory programming. *Computational Science and Engineering 5*, 1 (1998), 46–55.
9. Diaconescu, R.L. and Zima, H.P. An approach to data distributions in Chapel. Special issue on high-productivity programming languages and models. *International Journal of High Performance Computing Applications 21*, 3 (2007), 313–335.
10. Gupta, M., Schonberg, E., and Srinivasan, H. A unified framework for optimizing communication in data-parallel programs. *IEEE Transactions on Parallel and Distributed Systems 7*, 7 (July 1996), 689–704.
11. Hempel, R. and Walker, D.W. The emergence of the MPI message passing standard for parallel computing. *Computer Standards & Interfaces 21*, 1 (1999), 51–62.
12. High Performance Fortran Forum. High Performance Fortran language specification. *Scientific Programming 2*, 1–2 (1993), 1–170; see also CRPC–TR92225.
13. Hiranandani, S., Kennedy, K., and Tseng, C.-W. Compiling Fortran D for MIMD distributed-memory machines. *Commun. ACM 35*, 8 (Aug. 1992), 66–80.
14. Joisha, P.G. and Banerjee, P. The efficient computation of ownership sets in HPF. *IEEE Transactions on Parallel and Distributed Systems 12*, 8 (Aug. 2001), 769–788.
15. Kennedy, K., Koelbel, C., and Zima, H. The rise and fall of High Performance Fortran: An historical object lesson. In *Proceedings of the Third ACM SIGPLAN Conference on the History of Programming Languages* (San Diego). ACM Press, New York, 2007.
16. Kennedy, K. and Seo, Y., Eds. Concurrency and computation: Practice and experience. Special issue: *High Performance Fortran 14*, 8–9 (2002).
17. Koelbel, C., Mehrotra, P., Saltz, J., and Berryman, H. Parallel loop on distributed machines. In *Proceedings of the Fifth Distributed Memory Computing Conference* (Apr. 1990), 1097–1104.
18. Leasure, B. *Parallel Processing Model for High-Level Programming Languages.* Draft proposed American national standard for information processing systems, Apr. 1994.
19. Shimasaki, M. and Zima, H.P., Eds. The Earth Simulator, special edition. *Parallel Computing 30*, 12 (Dec. 2004).
20. Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J. *MPI: The Complete Reference: The MPI Core, Second Edition.* MIT Press, Cambridge, MA, 1998.
21. Sun Microsystems, Inc. *The Fortress Language Specification*, Version 0.707, Burlington, MA, July 2005.
22. Thinking Machines Corp. *CM Fortran Reference Manual, Version 1.0*, Cambridge, MA, Feb. 1991.

Ken Kennedy was the John and Ann Doerr Professor in the Department of Computer Science at Rice University, Houston, TX.

Charles Koelbel (chk@cs.rice.edu) is a research scientist in the Computer Science Department of Rice University, Houston, TX.

Hans Zima (zima@jpl.nasa.gov) is a principal scientist at the Jet Propulsion Laboratory, California Institute of Technology, and a professor emeritus at the University of Vienna, Austria.

DOI:10.1145/2018396.2018417

Technology able to create devices the size of a human cell calls for new protocols.

BY IAN F. AKYILDIZ, JOSEP MIQUEL JORNET, AND MASSIMILIANO PIEROBON

Nanonetworks: A New Frontier in Communications

IN 1959, THE Nobel laureate physicist Richard Feynman, in his famous speech entitled "There's Plenty of Room at the Bottom," described for the first time how the manipulation of individual atoms and molecules would give rise to more functional and powerful man-made devices. In his vision, he talked about having a billion tiny factories able to manufacture fully functional atomically precise nano-devices. During the same talk, he noted that several scaling issues would arise when reaching the nanoscale, which would require the engineering community to totally rethink the way in which nano-devices and nano-components are conceived.

More than a half-century later, current technological trends, which are still mainly based on the miniaturization of existing manufacturing techniques, are facing these predicted limitations. Consequently, there is a need to rethink and redesign the way in which components and devices are created by taking into account the new properties of the nanoscale. Moreover, a whole new range of applications can be enabled by the development of devices able to benefit from these nanoscale phenomena from the very beginning. These are the tasks at the core of the nanotechnology.

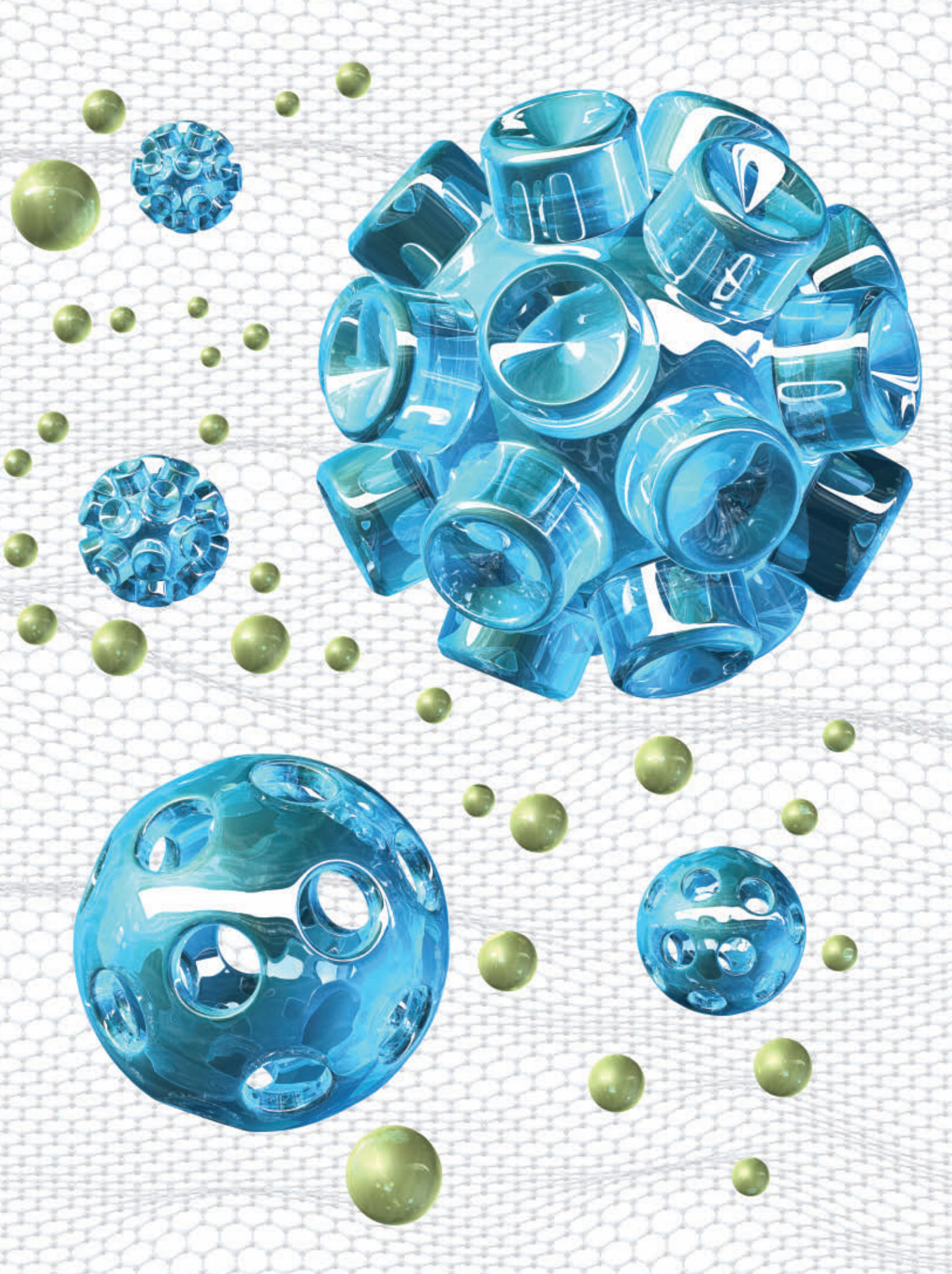
The term *nanotechnology* was first defined in a work dated from 1974[15] as follows: *"Nanotechnology mainly consists of the processing of, separation, consolidation, and deformation of materials by one atom or by one molecule."* Later, in the 1980s, the basic concept of this definition was explored in much more depth by K. Eric Drexler,[3] who took Feynman's vision about creating nano-devices by using tiny factories, and added the idea that they could replicate themselves via computer control instead. For more than 10 years, Drexler received numerous accusations of promoting science fiction. However, as the first simple structures on a molecular scale were obtained, the activities surrounding nanotechnology began to slowly increase and this term became more socially accepted. It was in the early 2000s when the major advancements in the field ramped up.

Among the different aims of nanotechnology, we focus on the development of *nanomachines*, that is, integrated functional devices consisting

» key insights

- **Nanotechnology is providing a new set of tools to the engineering community to design and manufacture nanomachines; that is, basic functional nano-devices able to perform only very simple tasks.**
- **Nanonetworks—networks of nanomachines—will expand the capabilities of single nanomachines by providing them a way to cooperate and share information.**
- **It is still not clear how nanomachines will communicate. Two main alternatives currently being considered are Terahertz Band and molecular communications.**

ILLUSTRATION BY GIACOMO MARCHESI



of nanoscale components and which are able to perform simple tasks at the nano-level. Going one step ahead, we propose the interconnection of nanomachines in a network or *nanonetwork* as the way to overcome the limitations of individual nano-devices.[1,2] The potential applications of the resulting nanonetworks are almost unlimited and can be classified in four main areas: *Biomedical Applications* (for example, intrabody health monitoring and drug delivery systems, immune system support mechanisms, and artificial bio-hybrid implants); *Industrial and Consumer Goods Applications* (for example, development of intelligent functionalized materials and fabrics, new manufacturing processes and distributed quality control procedures, food and water quality control systems); *Environmental Applications* (biological and chemical nanosensor networks for pollution control, biodegradation assistance, and animal and biodiversity control); and *Military Applications* (nuclear, biological and chemical defenses and nano-functionalized equipment).

Several communication paradigms can be used in nanonetworks depending on the technology used to manufacture the nanomachines and the targeted application. In this article, we provide an overview of the two main alternatives for nanocommunication, that is, Electromagnetic Communications in the Terahertz Band and Molecular Communications. Our aim is to provide a better understanding of the current research issues in this truly interdisciplinary and emerging field, and to pave the way of future research in nanonetworks. We also review the state of the art in the design and manufacturing of nanomachines, discuss the different alternatives for communication in the nanoscale, and describe the research challenges in the design of protocols for nanonetworks. While there is still a long way to go before a fully functional nanomachine is realized, we believe hardware-oriented research and communication-focused investigations will benefit from being conducted in parallel from an early stage.

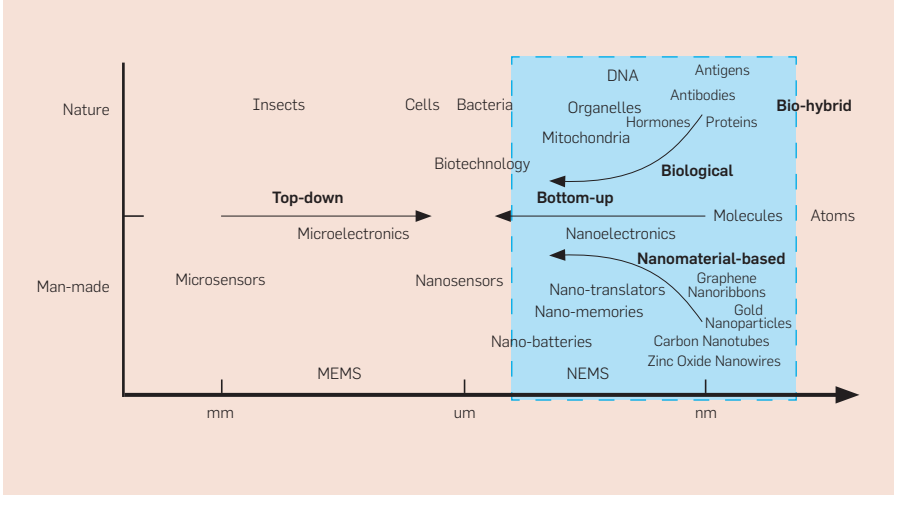
Manufacturing Nanomachines

Nanonetworks start at the interconnection of several nanomachines. The capabilities and the application range of these nanomachines strongly depend on the way in which they are manufactured. As shown in the accompanying figure, different approaches can be used for their development, ranging from the use of man-made components to the reuse of biological entities found in nature. These approaches are classified into three main branches, namely, top-down, bottom up and bio-hybrid.[1] In the *top-down* approach, nanomachines are developed by means of downscaling current microelectronic and micro-electro-mechanical technologies without atomic level control. In the bottom-up approach, the design of nano-machines is realized from the (self) assembly of molecular components and synthesized nanomaterials. Alternatively, in a *bio-hybrid* approach, existing biological components, such as Deoxyribonucleic Acid or DNA strands, antibodies or molecular motors, are combined with man-made nano-structures to develop new nanomachines.

Man-made machines. Despite several technological and physical limitations, the evolution of classical lithography techniques and other non-standard manufacturing procedures have been used to fabricate components with at least one of their dimensions in a scale below 100nm.[16] A special emphasis should be given to the study of nanomaterials and new manufacturing processes, which are enabling a new direction for the development of nano-components. As an example, field-effect transistors can be obtained through the use of graphene nanoribbons and carbon nanotubes, and these can be used as the building block for new computing machines.[14] Other well-studied nano-components are nanomaterials-based biological, chemical, and physical nanosensors and nanoactuators. The integration of several of these nano-components into a single functional unit will result in a device with a total size in between 10–100 square micrometers,[2] which is comparable to the size of an average human cell. However, the integration of these components into a single device is still one of the major challenges in the manufacturing of nanomachines.

Adopting components coming from nature. The nanoscale is the natural domain of molecules, proteins, DNA, organelles and the major components of cells. Some of these nano-components can be used as building blocks for integrated nano-devices. As an example, Adenosine TriPhosphate or ATP batteries emulating the behavior of *mitochondria*, often described as "cellular power plants," can be an alternative energy source for bio-nano-devices. In addition, information encoded in DNA can be used for molecular computing machines and molecular memories. Alternatively, DNA strands can also be used to build miniature circuit boards and to stimulate the self-assembly of components such as carbon nanotubes, nanowires, nanoribbons and

Approaches for the development of nanomachines.



nanoparticles, by means of DNA scaffolding.[9] While still being one step behind nanomaterial-based component manufacturing, we believe that being able to directly reuse biological structures found in living organisms or to reengineer them will be especially useful in biomedical applications, as well as the enabling technology for bio-inspired communications.

Enabling Communication Among Nanomachines

Nanocommunication is the exchange of information at the nanoscale and it is at the basis of any wired/wireless interconnection of nanomachines in a nanonetwork. The way in which the nanomachines can communicate depends strongly on the way they are realized. Moreover, the particular application for which the nanonetworks will be deployed constrains the choice on the particular type of nanocommunication. For the time being, several alternatives have been proposed. These range from downscaling well-established communication means based on electromagnetic, optical, acoustic, or mechanical communication, up to defining completely new paradigms inspired by biology.

Downscaling existing communication paradigms. The tools provided by nanotechnology are enabling the extension of well-known communication techniques to the nanoscale. First of all, carbon nanotubes and graphene nanoribbons have been proposed for electromagnetic nano-antennas.[6] A graphene-based nano-antenna is not just a mere reduction of a classical antenna, but there are several quantum phenomena that affect the propagation of electromagnetic waves on graphene. As a result, the resonant frequency of these nanostructures can be up to two orders of magnitude below that of their non-carbon-based counterparts. However, their radiation efficiency can also be impaired because of this phenomenon. Second, carbon nanotubes have also been proposed as the basis of an electromechanical nano-transceiver or nano-radio,[5] able to modulate and demodulate an electromagnetic wave by means of mechanical resonation. This technique has been experimentally proved in reception, but would require very high nanoscale power sources for active transmission.

Terahertz Band: Ultra-broadband communications in nanonetworks. Focusing on the use of graphene-based nano-antennas and thinking of the expected maximum size of a nanomachine, the Terahertz Band (0.1THz–10THz) enters the game. Indeed, we have recently shown that a one-micrometer-long graphene-based nano-antenna would expectedly resonate in the aforementioned band.[6] This very high-frequency range, in between the microwaves and the far-infrared radiation, has recently caught the attention of the scientific community because of its applications in security screening and nanoscale imaging systems. In our case, we think of the Terahertz Band as a very large transmission window that can support very high transmission rates in the short range, that is, up to a few Terabits per second for distances below one meter, or as several transmission windows more than 10 gigahertz-wide each as we've recently shown.[7] For the time being, it is not clear how nanomachines with limited capabilities can exploit the properties of this huge band, but several options come to mind. For example, we have recently proposed the use of very low energy femtosecond-long pulses as a simple but robust communication paradigm for nanomaterial-based nanomachines.[8] Moreover, having a very large available bandwidth introduces major changes in classical networking protocols, as we describe here.

Learning from biology: Molecular communication. Cells and many living organisms exchange information by means of molecular communication, that is, they use molecules to encode, transmit and receive information. Among others, one of the most widespread molecular communication mechanisms is based on the free diffusion of molecules in the space. For example, communication between neighboring cells in the human body is conducted by means of diffusion of different types of molecules, which encode different types of messages. To date, research has been carried out to study the propagation of molecular messages by means of free diffusion. Among others, in Pierobon and Akyildiz,[7] we analyzed the behavior of the molecular diffusion channel in terms of attenuation and delay. In the same paper, we provide mathematical models of the physical processes occurring at the molecular transmission, propagation and reception. The results of this work are in two different directions. First, they provide a numerical evaluation of the communication capabilities of the physical channel. Attenuation values of tens of dB for a transmission range up to 50 micrometers and a frequency up to 400Hz (but hundreds of dB when the frequency approaches 1kHz) have been obtained with a delay of more than 100ms. Second, the results define reliable and simple models, which can be used off the shelf in the design of molecular communication systems based on the free diffusion of molecules. We expanded our understanding of the molecular diffusion channel by analyzing the most relevant diffusion-based noise sources, whose origins are intrinsically different than for noise sources in EM communication.[13] Theoretical limits on the information capacity of a diffusion-based molecular communication system are studied in Pierobon and Akyildiz.[12] We show that the order of magnitude of the capacity for a molecular communication system is extremely higher than the capacity of classical communication systems. These results confirm the growing interest around molecular communication for nanonetworks shown by the research community in the last couple years.

Alternatively, in Parcerisa and Akyildiz,[10] we proposed the use of pheromones for molecular communication in long-range nanonetworks, such as, for transmission distances approximately one meter. Pheromones are molecules of chemical compounds released by plants, insects, and other animals that trigger specific behaviors among the receptor members of the same species and whose propagation relies also on the molecular diffusion process. In the same paper, we present other molecular communication techniques, such as neuron-based communication and capillaries flow circuits. The former refers to the possibility of building a communication

system directly inspired by the nerve fibers that transport muscle movements, external sensorial stimuli, and neural communication signals to and from the brain. The latter are inspired by the capillaries, which are the smallest blood vessels inside the human body. Capillaries connect arterioles and venules and their main function is to interchange chemicals and nutrients between the blood and the surrounding tissues. The feasibility and practicality of these systems still needs to be investigated, but they can serve as a starting point for future bio-inspired nanocommunication systems.

Last but not least, we proposed and studied in Gregori and Akyildiz[4] a molecule transport technique using two different types of carrier entities, namely, flagellated bacteria and catalytic nanomotors. On the one hand, the flagellated bacteria are able to carry DNA messages introduced inside their cytoplasm. When set free in the environment, the carrier bacteria are headed to the receiver, which is continuously releasing bacteria attractant particles. Upon contact with the receiver, the bacteria release the DNA message to the destination. On the other hand, the catalytic nanomotors are defined as particles that are able to propel themselves and small objects. Nanomotors can be loaded with DNA molecules and their propagation can be guided using preestablished magnetic paths from the emitter to the receiver. Nanomotors can also compose a raft and transport the DNA message through a chemotactic process. The propagation of information by means of guided bacteria or catalytic nanomotors is relatively very slow (in the order of a few millimeters per hour), but the amount of information that can be transmitted in a single DNA strand makes the achievable information rate relatively high (up to several kilobits per second). All these results require us to rethink well-established concepts in communication and network theory.

Developing Nanonetworking Protocols

Nanonetworks are systems composed by interconnected nanomachines that communicate by following specific protocols. These protocols must address various issues not only common to conventional networks, but also arising from the peculiarities of the different nanocommunication options. For this, several features required in any communication network, such as Medium Access Control (MAC) mechanisms, addressing schemes, or information routing techniques, must be designed in light of the properties of the aforementioned nanocommunication paradigms

While there is still a long way to go before a fully functional nanomachine is realized, we believe hardware-oriented research and communication-focused investigations will benefit from being conducted in parallel from an early stage.

Terahertz nanonetworks. The Terahertz Band provides a very large transmission bandwidth. On the one hand, this can be used to support very high-speed communication among nano-devices. On the other hand, a very large bandwidth enables new channel access techniques, which can ease the tasks of the MAC protocol. For example, when using femtosecond-long pulses for communication among nano-devices,[2] the chances of having a collision between different nanomachines' transmissions are almost nonexistent. As a result, very simple MAC protocols can be used. For instance, nanomachines can just transmit whenever they have some information ready and then they just wait for an acknowledgment. New ways to verify the integrity of the message that has been transmitted and to accordingly inform the transmitter will be necessary. In addition, in light of the capabilities of nanomachines, new coding schemes and error correction mechanisms will have to be developed. When it comes to addressing and routing, it will also be the capabilities of nanomachines what will determine what is possible and what is not. For example, it seems unfeasible to assign a unique ID to every component of a nanonetwork. Alternatively, by exploiting again the nature of pulse-based communications, we think that nanomachines will have a notion of the distances among them, which can be used for addressing and routing purposes. At the same time, in our vision, we believe that in some applications it will not be necessary to uniquely identify every nanomachine, but it will be enough by just classifying the nanomachines according to their internal status, for example, the sensing readings.

Molecular nanonetworks. Molecu-

lar nanonetworks require the development of new networking protocols suited for the nature of this new paradigm. In our vision, the study of the molecular network protocols will follow a twofold approach: on the one hand, network structures and protocols will be directly inspired from the observation of communication and signaling processes from nature (biologically inspired molecular networks); on the other hand, classical networking paradigms will be adapted for their use with synthesized molecular nanonetworks. In both of the two cases, these protocols will have to take into account the delay in the propagation of molecular information, which is considerably high if compared to electromagnetic communications. The study of a MAC protocol should also take into account the effects of the interaction of multiple molecular transmitters in the same environment. The performance of the molecular communication system in terms of attenuation and delay will likely vary due to the interactions physically occurring between molecules emitted by different transmitters, such as collisions or electrical and chemical reactions. Therefore, a MAC protocol will be required to minimize the interference between different emitters and to maximize the overall throughput of the network. Moreover, routing and addressing aspects will be required to enable communication between multiple source and destination points. In our vision, any form of addressing will be likely embedded within the structure of the molecules that compose the information message, such as their type or even electrical charge. Molecular protocols will also be studied in relation to the particular adopted molecular communication technique. As an example, when pheromones are used as information carriers, routing protocols will have to take into account the fact that their propagation in the air medium is highly dependent on the direction of the wind flow. Particular geographical routing algorithms could exploit the knowledge of the current and future direction of the wind to achieve a direction-based addressing. Another example is given by the flagellated bacteria communication, where addressing can be achieved by engineering bacteria able to sense only some types of attractants which are released only by the targeted receivers.

Conclusion

Nanonetworks will have a great impact in almost every field of our society ranging from health care to homeland security and environmental protection. In order to enable the communication among nanomachines, it is necessary to rethink existing communication paradigms and to define new communication alternatives stemming from the nature of the nanoscale. While the hardware underlying nanonetworks is still being developed, the engineering of new computing and data storage architectures for nanoscale devices, the definition of new information encoding and modulation for nanomachines using different nanocommunication paradigms, and the development of nanonetworking structures and protocols are necessary contributions expected from the ICT field.

Acknowledgment

This work was supported by the U.S. National Science Foundation (NSF) under Grant No. CNS-1110947 and Fundación Caja Madrid.

References

1. Akyildiz, I., Brunetti, F. and Blazquez, C. Nanonetworks: A new communication paradigm. *Computer Networks Journal 52*, 12, (Aug. 2008), Elsevier, 2260–2279.
2. Akyildiz, I.F. and Jornet, J.M. Electromagnetic wireless nanosensor networks. *Nano Communication Networks Journal 1*, 1 (Mar. 2010), Elsevier, 3–19.
3. Drexler, E. Molecular engineering: Assemblers and future space hardware. *American Astronautical Society*, 1986.
4. Gregori, M. and Akyildiz, I.F. A new NanoNetwork architecture using flagellated bacteria and catalytic nanomotors. *IEEE Journal of Selected Areas in Communications 28*, 4 (May 2010), 612–619.
5. Jensen, K., Weldon, J., Garcia, H. and Zettl, A. Nanotube radio. *Nano Letters 7*, 11 (Nov. 2007), 3508–3511.
6. Jornet, J.M. and Akyildiz, I.F. Graphene-based nano-antennas for electromagnetic nanocommunications in the terahertz band. In *Proceedings of 4th European Conference on Antennas and Propagation* (Barcelona, Spain, 2010), 1–5.
7. Jornet, J.M. and Akyildiz, I.F. Channel modeling and capacity for electromagnetic wireless nanonetworks in the Terahertz Band. To appear in *IEEE Transactions on Wireless Communications*, 2011.
8. Jornet, J.M. and Akyildiz, I.F.Information capacity of pulse-based wireless nanosensor networks. In *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh, and Ad Hoc Communications and Networks* (Salt Lake City, UT, June 2011)
9. Kershner, R.J. et al. Placement and orientation of individual DNA shapes on lithographically patterned surfaces. *Nature Nanotechnology 4*, 9 (Aug. 2009), 557–561.
10. Parcerisa, L. and Akyildiz, I.F. Molecular communication options for long range nanonetworks. *Computer Networks Journal 53*, 16 (Nov. 2009), Elsevier, 2753–2766.
11. Pierobon, M. and Akyildiz, I.F. A physical end-to-end model for molecular communication in nanonetworks. *IEEE Journal of Selected Areas in Communications 28*, 4 (May 2010), 602–611.
12. Pierobon, M. and Akyildiz, I.F. Information capacity of diffusion-based molecular communication in nanonetworks. In Proceedings of the *IEEE International Conference on Computer Communication* (Apr. 2011 miniconference).
13. Pierobon, M. and Akyildiz, I.F. Diffusion-based noise analysis for molecular communicaion in nanonetworks. *IEEE Transactions on Signal Processing 59*, 6 (June 2011), 2532–2547.
14. Ponomarenko, L.A. et al. Chaotic dirac billiard in graphene quantum dots. *Science 320*, 5874 (Apr. 2008), 356–358.
15. Taniguchi, N. On the basic concept of nano-technology. In *Proceeding of the International Conference on Production Engineering*, 1974.
16. Wang ,Y., Mirkin, C.A. and Park S. J. Nanofabrication beyond electronics. *ACS Nano 3*, 5 (2009), 1049–1056.

Ian F. Akyildiz (ian@ece.gatech.edu) is the Ken Byers Chair Professor in Telecommunication, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, and director of the Broadband Wireless Networking Laboratory.

Josep Miquel Jornet (jmjornet@ece.gatech.edu) is a Ph.D. student at Broadband Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta.

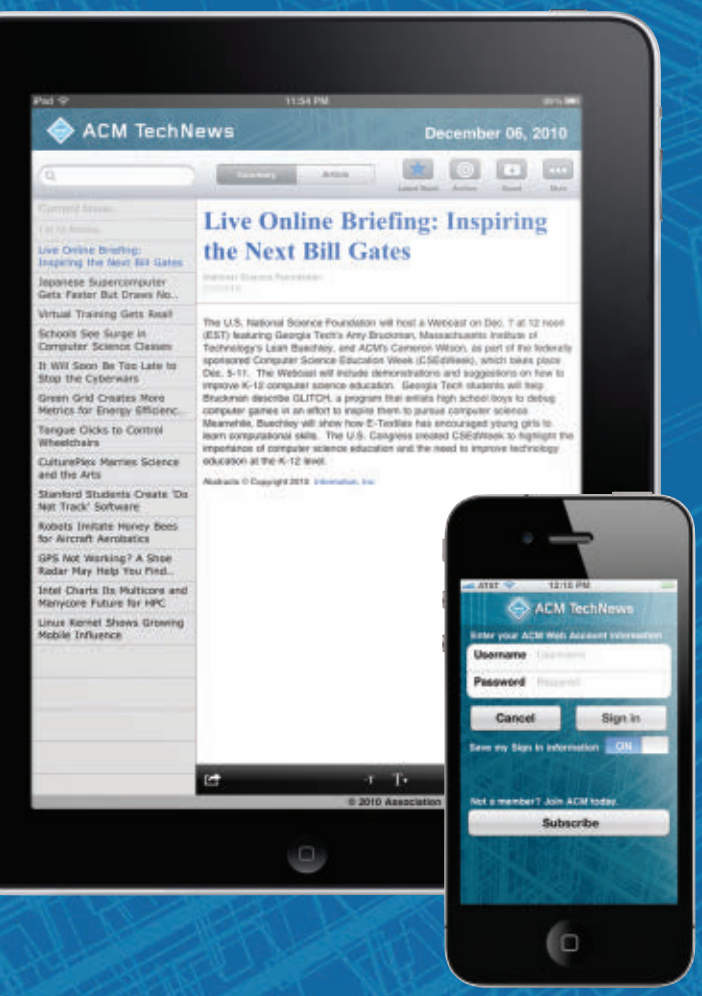
Massimiliano Pierobon (maxp@ece.gatech.edu) is a Ph.D. student at Broadband Wireless Networking Laboratory, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta.

ACM TechNews Goes Mobile

iPhone & iPad Apps Now Available in the iTunes Store

ACM TechNews—ACM's popular thrice-weekly news briefing service—is now available as an easy to use mobile apps downloadable from the Apple iTunes Store.

These new apps allow nearly 100,000 ACM members to keep current with news, trends, and timely information impacting the global IT and Computing communities each day.



TechNews mobile app users will enjoy:

- ***Latest News:*** Concise summaries of the most relevant news impacting the computing world
- ***Original Sources:*** Links to the full-length articles published in over 3,000 news sources
- ***Archive access:*** Access to the complete archive of TechNews issues dating back to the first issue published in December 1999
- ***Article Sharing:*** The ability to share news with friends and colleagues via email, text messaging, and popular social networking sites
- ***Touch Screen Navigation:*** Find news articles quickly and easily with a streamlined, fingertip scroll bar
- ***Search:*** Simple search the entire TechNews archive by keyword, author, or title
- ***Save:*** One-click saving of latest news or archived summaries in a personal binder for easy access
- ***Automatic Updates:*** By entering and saving your ACM Web Account login information, the apps will automatically update with the latest issues of TechNews published every Monday, Wednesday, and Friday

The Apps are freely available to download from the Apple iTunes Store, but users must be registered individual members of ACM with valid Web Accounts to receive regularly updated content.

http://www.apple.com/iphone/apps-for-iphone/ **http://www.apple.com/ipad/apps-for-ipad/**

ACM TechNews

research highlights

DOI:10.1145/2018396.2018418

Technical Perspective
Making Untrusted Code Useful

By Butler Lampson

THE FOLLOWING PAPER combines two important themes in secure computing: *assurance* and information *flow control*. Assurance is evidence that a computer system is secure, that is, obeys its security specification, usually called a security *policy*. A system's Trusted Computing Base (TCB) is all the parts that must work for the system to be secure. For high assurance, the TCB needs to be small and the policy simple. Flow control is one such policy; it specifies how information is allowed to move around in a system.

In the late 1960s the military began to worry about a "multilevel" computer system in which secrets leak from a user process handling classified data. It is impractical to require every application program to come from a trusted source; databases, math libraries, and many other essential tools are too big and complicated to rebuild, or even to audit. So the secret process might contain a Trojan horse from the KGB, which leaks the secrets to Moscow via another, unclassified process.

Flow control solves this problem by requiring that no action of a secret process can affect the state of an unclassified one. Formally, it models the system as a state machine; the state is a set of variables (including process state such as the program counter), and every step sets some variables to functions of others. So each step has the form

$$\text{var}_1, \ldots \text{var}_n := f_1(\text{args}_1), \ldots, f_n(\text{args}_n)$$

The only flows are from args_i to var_i.

Each variable v has a *label* $L(v)$. Labels form a lattice, a partial order $\sqsubseteq$ in which any two elements have a least upper bound or max. The flow rule is that information can only flow from weaker (unclassified) labels to stronger (secret) ones, so each step must satisfy

$$\max_{v \in \text{args}_i} L(v) \sqsubseteq L(\text{var}_i)$$

Typically a label is a set of atomic elements called *categories*, the ordering is set inclusion, and max is set union.

The flow rule is good because it composes: if each step obeys the rule, the whole computation does so. Hence the label on every data item is at least the max of the labels on everything that affected it; the rule is end to end. It is certainly simple, and assurance is just evidence that each step obeys it.

In the early 1980s research on flow led to the "Orange Book," which defines the security of a computer system by how well it implements flow control and how good its assurance is. The government said that it would require all multilevel systems to be secure in this sense, and several vendors developed them. Sadly, they all turned out to have rather large TCBs and to be slow, clumsy to use, and years behind less secure systems in functionality. The requirement was frequently waived, and it was finally abandoned. After this discouraging experience people lost interest in information flow, especially since a personal computer is not usually multilevel. A networked computer is always multilevel, however, and today all computers are networked (though most adversaries are much weaker than the KGB).

Ten years ago Myers and Liskov[2] revived the field by pointing out that categories need not be predefined. Instead, any process can create a new category, which it owns. An owner can declassify a category, that is, remove it from a label. This is appealingly decentralized and Internet friendly, and makes it possible to *isolate* any program by running it with a new category in its label and then declassifying the result; the flow rule ensures there are no other visible effects.

This paper describes HiStar, a system that enforces decentralized information flow directly. The variables are OS objects, files, threads, etc., rather than integers. Security depends on the simple flow control policy and a small TCB, a new 2,000 line kernel with six object types, each with just a few methods. This is much less than an existing code base retrofitted for strong security. HiStar implements access control using control flow, the reverse of previous practice. Flows through shared resources, such as using up a resource or deleting an object, are tricky problems whose solutions add complexity. Containers enable sharing by holding hard links to objects. Unreachable objects are garbage collected, so there is no deletion.

Security depends on the simple flow control policy and a small TCB.

Unix applications run on a library OS[1] that is not part of the TCB, using containers to represent directories, file descriptors, and other protected state. It's surprising that such a minimal kernel suffices, but similar results have recently been reported for Windows.[3] Isolation can clone a whole Unix; a container argument provides the environment: the file system root, address space, resources, and so on. The clone can run most Unix programs, and it can't affect anything outside of itself.

HiStar works well for tasks in which untrusted code reads sensitive data, such as setting up an SSL connection or running a virus scanner. It can isolate a lot of untrusted code using a little trusted code as a wrapper that decides how to declassify the results. The general-purpose computing that failed in the 1980s has not been tried.

This is the latest step in the long and frustrating journey toward secure computing. It is a convincing solution for some serious practical problems. Of course the devil is in the details, which you can read about in the paper. C

References

1. Kaashoek, F. et al. Application performance and flexibility on exokernel systems. *ACM Operating Systems Review 31*, 5, (Dec. 1997), 52–65.
2. Myers, A. and Liskov, B. Protecting privacy using the decentralized label model. *Trans. Comput. Syst. 9*, 4 (Oct. 2000), 410–442.
3. Porter, D. et al. Rethinking the library OS from the top down. *ACM SIGPLAN Notices 46*, 3 (Mar. 2011), 291–304.

Butler Lampson (Butler.Lampson@microsoft.com) is a Technical Fellow at Microsoft Research and is a Fellow of ACM.

DOI:10.1145/2018396.2018419

Making Information Flow Explicit in HiStar

By Nickolai Zeldovich, Silas Boyd-Wickizer, Eddie Kohler, and David Mazières

Abstract

HiStar is a new operating system designed to minimize the amount of code that must be trusted. HiStar provides strict information flow control, which allows users to specify precise data security policies without unduly limiting the structure of applications. HiStar's security features make it possible to implement a Unix-like environment with acceptable performance almost entirely in an untrusted user-level library. The system has no notion of superuser and no fully trusted code other than the kernel. HiStar's features permit several novel applications, including privacy-preserving, untrusted virus scanners and a dynamic Web server with only a few thousand lines of trusted code.

1. INTRODUCTION

Many serious security breaches stem from vulnerabilities in application software. Despite an extensive body of research in preventing, detecting, and mitigating the effects of software bugs, the security of most systems ultimately depends on a large fraction of the code behaving correctly. Unfortunately, experience has shown that only a handful of programmers have the right mind-set to write secure code, and few applications have the luxury of being written by such programmers. As a result, we see a steady stream of high-profile security incidents.

How can we build secure systems when we cannot trust programmers to write secure code? One hope is to separate the security critical portions of an application from the untrusted bulk of its implementation; if security depends on only a small amount of code, this code can be verified or implemented by trustworthy parties regardless of the complexity of the application as a whole. Unfortunately, traditional operating systems do not lend themselves to such a division: they make it too difficult to predict the full implications of every action by untrusted code.[7] HiStar is a new operating system designed to overcome this limitation.

HiStar enforces security by controlling how information flows through the system. Hence, one can reason about which components of a system may affect which others and how, without having to understand those components themselves. Specifying policies in terms of information flow is often much easier than reasoning about the security implications of individual operations.

As an example, let us consider anti-virus software, which often has full access to all files on a user's computer. There have been critical vulnerabilities discovered in virus scanners from Norton,[14] McAfee,[10] and others[15] that allow attackers to take full control of the scanner. Such vulnerabilities can easily be exploited to, at the very least, steal private data from millions of users. To prevent such a disaster, we might switch to the simpler, open-source ClamAV virus scanner. However, it has suffered from security vulnerabilities in the past,[21] and is over 40,000 lines of code—large enough that hand-auditing the system to eliminate vulnerabilities would be an expensive and lengthy process at best. Yet a virus scanner must periodically be updated on short notice to counter new threats, in which case users would face the unfortunate choice of running either an outdated virus scanner or an unaudited one. A better solution would be for the operating system to enforce security without trusting ClamAV to keep the user's data private, thereby minimizing potential damage from ClamAV's vulnerabilities.

Figure 1 illustrates ClamAV's components. How can we protect a system should these components be compromised? Among other things, we must ensure a compromised ClamAV cannot purloin private data from the files it scans, or corrupt those files. In doing so, we must also avoid imposing restrictions that might interfere with ClamAV's proper operation—for example, the scanner needs to spawn a wide variety of external helper programs to decode input files. Here are just a few ways in which, on Linux, a maliciously controlled scanner and update daemon can collude to copy private data to an attacker's machine:

- The scanner can send the data directly to the destination host over a TCP connection.
- The scanner can trick another program, such as a mail server running on the same machine, into transmitting the data.
- The scanner can take over an existing process using debug mechanisms (e.g., *ptrace* on Unix), and send the data via that process.
- The scanner can write the data to a file in `/tmp`. The update daemon can then read the file and leak the data by encoding it in the contents, ordering, or timing of subsequent network packets.
- The scanner can use any number of less efficient and subtler techniques to impart the data to the update daemon—for example, use file locking to lock different ranges of the database, bind particular TCP or UDP port numbers, modulate memory or disk usage in a detectable way, or change the title of the scanner process.

This work was originally presented at the 7th Symposium on Operating Systems Design and Implementation and the 5th Symposium on Networked Systems Design and Implementation.

Figure 1. The ClamAV virus scanner. Circles represent processes, rectangles represent files and directories, and rounded rectangles represent devices. Arrows represent the expected data flow for a well-behaved virus scanner.

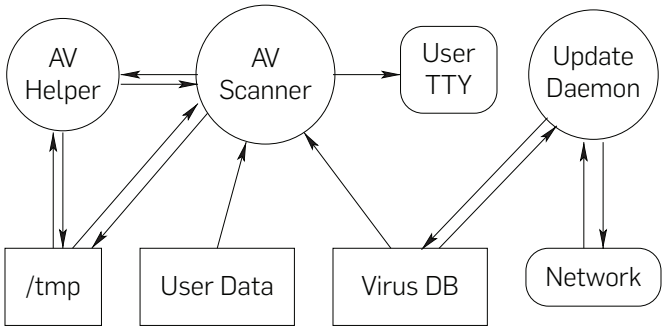
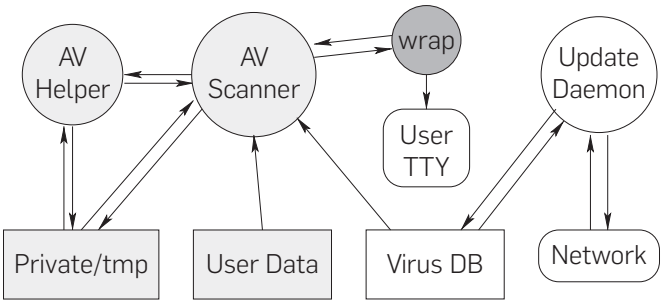


Figure 2. ClamAV running in HiStar. Lightly shaded components are *tainted*, which prevents them from conveying any information to untainted (unshaded) components. The strongly shaded `wrap` has untainting privileges, allowing it to relay the scanner's output to the terminal.



Some of these attacks can be mitigated by running the scanner with its own user ID in a *chroot* jail.[7] However, doing so requires highly privileged, application-specific code to set up the *chroot* environment, and risks breaking the scanner or one of its helper programs due to missing files.[7] Other attacks, such as those involving sockets or System V IPC, can be prevented only by modifying the kernel to restrict certain system calls. Unfortunately, devising an appropriate policy in terms of system call arguments is an error-prone task, which, if incorrectly done, risks leaking private data or interfering with operation of a legitimate scanner.

A better way to specify the desired policy is in terms of where information should flow—namely, along the arrows in the figure. While Linux cannot enforce such a policy, HiStar can. Figure 2 shows our port of ClamAV to HiStar. There are two differences from Linux. First, we have labeled files with private user data as *tainted*. Tainting a file restricts the flow of its contents to any untainted component, including the network. The second difference from Linux is that we have launched the scanner from a new, 110-line program called *wrap*, which has *untainting* privileges. *Wrap* untaints the virus scanner's result and reports back to the user. The scanner cannot read tainted user files without first tainting itself. Once tainted, it can no longer convey information to the network or update daemon. As long as *wrap* is correctly implemented, ClamAV cannot leak the contents of the files it scans.

Although HiStar's tainting mechanism appears simple at a high level, making it work in practice requires addressing a number of challenges. First, there are myriad ways in which data can leak out onto the network, as illustrated above with Linux. How would an operating system like HiStar know to check the taint of the data being leaked for each and every one of them? Second, a typical OS kernel already provides a wide range of protection mechanisms, including user IDs, process memory protection, *chroot* jails, and so on. How can we avoid further complicating the kernel with yet another mechanism, or at very least, avoid unexpected interactions between the many disparate protection mechanisms? Finally, managing the tainting of files and the untainting privileges requires a separate mechanism, which can equally well be the target of attacks. One answer is to allow only the system administrator—root—access to this mechanism, but doing so both hampers the ability of other applications to use this mechanism and increases the amount of fully privileged code running as root.

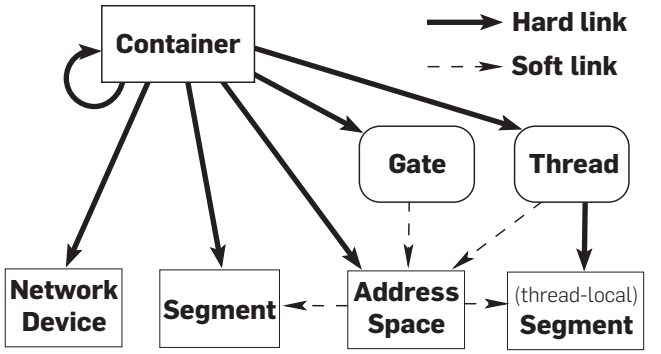
HiStar addresses these challenges with three key ideas. First, instead of implementing a traditional Unix interface, the kernel provides a lower-level interface, consisting of six types of kernel objects and a small number of operations that make any information flows between objects *explicit*. This provides a correspondingly small number of places where the kernel must perform data flow checks. Second, the *only* protection mechanism provided by the kernel is an information flow control mechanism, which generalizes the intuition behind taint. All other forms of protection, including Unix user IDs, process memory protection, and tainting itself, are implemented *in terms of* information flow control. This both reduces the amount of trusted kernel code and avoids any ambiguity about how the mechanisms will form a coherent policy. Finally, HiStar's information flow control mechanism is egalitarian, meaning that it can be used by any process, not just by superuser, which further reduces the amount of fully trusted code.

Though we used the virus scanner as an example, many security problems can be couched in terms of information flow. For example, protecting users' private profiles on a Web site often boils down to ensuring one person's information (Social Security number, credit card, etc.) cannot be sent to another user's browser. Protecting against trojan horses means ensuring network payloads do not affect the contents of system files. Protecting passwords means ensuring that whatever code verifies them can reveal only the single bit signifying whether or not authentication succeeded. The rest of this paper describes how HiStar provides a new, Unix-like environment in which small amounts of code can secure much larger, untrusted applications by enforcing such policies.

2. DESIGN

The HiStar kernel is organized around six object types, shown in Figure 3: a *segment* (a variable-length byte array similar to a file), an *address space* (a mapping from virtual memory addresses to segment object names), a *network*

Figure 3. Kernel object types in HiStar. *Soft links* name objects by a particular ⟨container ID, object ID⟩ container entry. Threads and gates are represented by rounded rectangles to indicate they are the only objects that have ownership privileges.



device (which can send and receive packets), a *thread* (a set of CPU registers, along with the name of an address space object), a *gate* (an IPC mechanism), and a *container* (a directory-like object in which all other objects reside).

Each object has a unique 64-bit *object ID* and a *label* that is used to control information flow to or from that object. All of the state accessible to user processes is stored in kernel objects (except for a few global variables, such as the counter used to allocate fresh object IDs). Thus, to read or write any data, processes must invoke the kernel (e.g., issue a system call or trigger a page fault to access a memory-mapped file). Upon receiving such a request, the kernel compares the labels of the currently executing thread and the objects being accessed to decide whether the operation should be permitted. While it is not possible to interpose on every read and write to memory-mapped files, the kernel remembers all active memory mappings and invalidates them when it suspects access should no longer be allowed (e.g., when a thread's label changes).

2.1. Labels

Before discussing the kernel interface further, we first describe HiStar's labels more precisely. HiStar associates a label with every kernel object. The purpose of a label is to provide a conservative estimate of what kind of data might be present in an object.

Generalizing the virus-scanner example shown in Figure 2, there may be multiple kinds of secret data in a system, perhaps belonging to different users. HiStar uses the notion of secrecy *categories* to distinguish between different kinds of secret data (a category is just an opaque 64-bit identifier), and a label is simply a set of categories. For example, in Figure 2, lightly shaded components have one specific secrecy category in their label; processes and files not shown in the figure can be labeled with one or more other categories. Data can flow from object A to object B only if B's label includes all of the secrecy categories in A's label. This is similar to the Bell-LaPadula model,[1] and ensures that any data marked secret remains in objects marked secret.

Secrecy categories help control where secret data can end up, but it is also important to control where data comes from. For instance, the virus scanner may want to ensure its virus database has not been corrupted by another application, and one user may want to prevent other users from overwriting his files. To address this problem, HiStar provides a second type of category—an *integrity* category. The type of a category is stored in the high bit of the category's 64-bit identifier, but in the rest of this paper we will use the notation c_r to indicate a secrecy (read) category and c_w to indicate an integrity (write) category. Object labels can include both secrecy and integrity categories, but the rules for integrity categories are the opposite of secrecy: data can flow from A to B only if A's label includes all of the integrity categories in B's label. This is analogous to the Biba integrity model,[2] and ensures that high-integrity files can be modified only by high-integrity sources.

Given these two types of categories, we can formalize when data can flow between two objects. For any two objects A and B, with labels L_A and L_B, data can flow from A to B if and only if every secrecy category in L_A is present in L_B, and every integrity category in L_B is present in L_A. This relation is checked frequently by HiStar, and we denote it by $L_A \sqsubseteq L_B$ (pronounced L_A *can flow to* L_B). Note that the $\sqsubseteq$ relation is transitive, meaning that one can understand if data can flow between two objects without having to consider all possible intermediate objects through which the data may flow.

While these rules ensure that secret data can propagate only to secret-labeled objects, a practical system requires occasionally extracting secret data from the system. For example, the *wrap* program shown in Figure 2 needs to send the output of the virus scanner to the user's terminal. HiStar allows this using the notion of category *ownership*. Each thread T, in addition to having a label L_T, owns a set of categories O_T, and these categories are ignored when performing operations on behalf of T. For example, T can read object A if $L_A - O_T \sqsubseteq L_T - O_T$, which we write as $L_A \sqsubseteq_{O_T} L_T$ (pronounced L_A *can flow using privileges* O_T *to* L_T).

In our virus scanner example, a user's files could be labeled $L_f = \{u_r, u_w\}$, where u_r and u_w are categories owned by the user that protect the secrecy and integrity of that user's data. The virus scanner runs with label $L_s = \{u_r\}$ and empty ownership set $O_s = \varnothing$, which allows it to read the user's files ($L_f \sqsubseteq_{O_s} L_s$), but not to modify them ($L_s \not\sqsubseteq_{O_s} L_f$) or export them (since the network is labeled $\varnothing$ and $L_s \not\sqsubseteq_{O_s} \varnothing$). The *wrap* process has label $L_w = \varnothing$ and ownership set $O_w = \{u_r\}$, which allows it to read data from the scanner ($L_s \sqsubseteq_{O_w} L_w$) and write it to the user's terminal ($L_w \sqsubseteq_{O_w} \varnothing$). Typically, a process trusted by the user owns both u_r and u_w, giving it the privileges to read, write, and export that user's data.

A key property of HiStar's labels is that ownership of one category confers no privileges with respect to other categories. This means that, for any secrecy category c_r, data labeled with c_r will flow only to objects labeled with c_r, unless a thread that owns c_r intervenes, and vice versa for integrity. This makes it possible to provide end-to-end guarantees on how different components can affect each other, by just inspecting components that own the relevant categories. For example, in Figure 2, it suffices to examine *wrap* to understand how shaded components can affect unshaded ones.

2.2. Labeling kernel state

HiStar's kernel enforces information flow control by

associating a label with every piece of user-visible state in the system—such as the registers of a thread, the length of a segment, and even the label of an object itself—and using the $\sqsubseteq$ relation to decide if a given thread should be allowed to observe or modify that state. As long as every piece of kernel state that can, directly or indirectly, influence execution of user code has a consistent label, then $\sqsubseteq$'s transitivity guarantees security: if data from thread A can affect some piece of kernel state X, and data from X can flow to some other thread B, then we must have checked that $L_A \sqsubseteq L_X$ and $L_X \sqsubseteq L_B$, which implies $L_A \sqsubseteq L_B$, so it was safe for data to flow from A to B. The key to ensuring transitivity lies in associating a consistent label with each piece of kernel state regardless of how the user code tries to, directly or indirectly, learn its value or modify it.

The bulk of HiStar's kernel state resides in kernel objects. For example, the simplest type of object is a segment, which contains a variable-length byte array. When thread T attempts to read segment S, either by issuing a system call or by triggering a page fault, the kernel checks that $L_S \sqsubseteq_{O_T} L_T$. Likewise, when thread T attempts to write S, the kernel checks that $L_T \sqsubseteq_{O_T} L_S \sqsubseteq_{O_T} L_T$. (The kernel ensures that data is allowed to flow from T to S and vice versa; our experience suggests it is difficult to write to an object without receiving some information as to whether the write succeeded.) As another example, a network device object's payload is (logically) all of the packets on the Ethernet network. To send or receive a packet, a thread must be able to write or read the network device, respectively, with rules identical to those for a segment.

Each object also contains the object's ID, the label, and a 64-byte mutable, user-defined metadata buffer (used by user-level code to, for instance, track modification times). The metadata buffer can logically be thought of as part of the mutable object contents, and is subject to the same read and write rules as the object contents. On the other hand, the label of an object O, L_O, presents a challenge: how should we label L_O's bytes? Suppose that we used L_O as the label of the entire object O, including L_O itself. If a thread tries to read O and is denied access, it learns something about the contents of L_O, even though this flow was prohibited.

To solve this chicken-and-egg problem, HiStar logically associates O's parent container's label with the bytes comprising L_O (as a special case, the root container is its own parent). Furthermore, because O might reside in multiple parent containers, HiStar requires that object labels be specified at creation and then immutable (except for threads, as we discuss later).

To deal with on-disk state, HiStar provides a single-level store: on bootup, the entire system (including threads) is restored from the most recent on-disk snapshot. This eliminates the need for trusted boot scripts to reinitialize processes that would not survive a reboot on traditional operating systems. It also achieves economy of mechanism by allowing the file system to be implemented with the same kernel abstractions as virtual memory, without any additional mechanisms for labeling on-disk state.

Finally, the kernel maintains a small amount of state outside of kernel objects, namely, the counter used to generate new object and category IDs. Newly allocated IDs must have two properties: first, they must be unique, and second, they must disclose almost no information about the state of the system, such as the number of previously allocated objects (*almost* because by definition, a new ID reveals the fact that this exact ID value was never allocated before). HiStar generates IDs by encrypting a counter with a block cipher. Since the block cipher is a pseudo-random one-way function, an attacker cannot learn any information from the value of the ID itself, and since the block cipher is a permutation, the IDs are unique.

2.3. Threads

Each thread T has a label L_T and an ownership set O_T, which can be changed through two mechanisms. First, a thread can allocate a fresh category by invoking the system call

- cat_t *create_category* (cat_type t),

which chooses a previously unused category, c, and adds c to O_T. The type of the category (secrecy or integrity) is specified by t. At this point, T is the only thread that owns c, and since c was never used before, granting T ownership of c confers no other privileges. In this sense, labels are egalitarian: no thread has any inherent privileges with respect to categories created by other threads. T can also drop categories from its ownership set.

T may change its own label through the system call

- int *self_set_label* (label_t L),

which sets $L_T \leftarrow L$, as long as $L_T \sqsubseteq_{O_T} L$. This can, for example, let T read a tainted object, or to untaint its label in categories it owns. HiStar also includes a clearance mechanism,[18] which prevents a thread from arbitrarily raising its label to read all possible data, but its discussion is omitted here for clarity.

A thread T can allocate new objects with label L as long as $L_T \sqsubseteq_{O_T} L$. Threads and gates (which will be discussed shortly) can be created with an ownership set O as long as $O \subseteq O_T$.

2.4. Containers

Because HiStar has no notion of superuser yet allows any software to create protection domains, nothing prevents a buggy thread from allocating resources in some new, unobservable, unmodifiable protection domain. To ensure that such resources can nonetheless be reclaimed, HiStar provides hierarchical control over object allocation and deallocation through *containers*. Like Unix directories, containers hold *hard links* to objects. There is a specially designated root container, which can never be deallocated. Any other object is deallocated once there is no path to it from the root container. Figure 3 shows the possible links between containers and other types of objects.

When allocating an object, a thread must specify the container into which to place the object. For example, to create a container, thread T makes the system call

- id_t *container_create* (id_t C, label_t L).

Here C is the object ID of an existing container, into which the newly created container will be placed. L is the desired label for the new container. The system call succeeds only if T can write to C (i.e., $L_T \sqsubseteq_{O_T} L_C \sqsubseteq_{O_T} L_T$) and allocate an object of label L

(i.e., $L_T \sqsubseteq_{O_T} L$). Objects can be likewise *unreferenced* from container C by any thread that can write to C. When an object has no more references, the kernel deallocates it. Unreferencing a container causes the kernel to recursively unreference the entire subtree of objects rooted at that container.

Containers also help HiStar address a possible covert channel through object reference counting. Any thread T can create a hard link to segment S in container C if it can write C (i.e., $L_T \sqsubseteq_{O_T} L_C \sqsubseteq_{O_T} L_T$). T can thus prolong S's life even without permission to modify S; in our virus-scanner example from Figure 2, this might be the malicious scanner process signaling secret information by prolonging or not prolonging the life of the database file. Another thread T', such as the update process, could then remove any known links to S and observe whether it can still access S by its object ID, even if T was not allowed to communicate to T'.

To avoid this problem, most system calls name objects by ⟨container ID, object ID⟩ pairs, called *container entries*. For T' to use container entry $\langle C, S\rangle$, C must contain a link to S and T' must be able to read C (i.e., $L_C \sqsubseteq_{O_{T'}} L_{T'}$). In the virus-scanner example, the untainted update process would not be able to use any container entry created by the tainted scanner. Container entries allow the kernel to check if a thread has permission to know if the object exists, in addition to any other label checks necessary to access the object.

2.5. Address spaces

Every running thread has an associated address space object containing a list of VA → ⟨*S*, *offset*, *npages*, *flags*⟩ mappings. VA is a page-aligned virtual address. $S = \langle C, O\rangle$ is a container entry for a segment to be mapped at VA. *offset* and *npages* can specify a subset of S to be mapped, *flags* specifies read, write, and execute permission (and some convenience bits for user-level software).

Each address space A has a label L_A, to which the usual label rules apply. Thread T can modify A only if $L_T \sqsubseteq_{O_T} L_A \sqsubseteq_{O_T} L_T$, and can observe or use A only if $L_A \sqsubseteq_{O_T} L_T$. When launching a new thread, one must specify its address space and program counter. The system call *self_set_as* allows threads to switch address spaces. When thread T takes a page fault, the kernel looks up the faulting address in T's address space to find a segment $S = \langle C, O\rangle$ and *flags*. If *flags* allows the access mode, the kernel checks that T can read C and O ($L_C \sqsubseteq_{O_T} L_T$ and $L_O \sqsubseteq_{O_T} L_T$). If *flags* includes writing, the kernel additionally checks that T can modify O ($L_T \sqsubseteq_{O_T} L_O$). If no mapping is found or any check fails, the kernel calls up to a user-mode page-fault handler (which by default kills the process). If the page-fault handler cannot be invoked, the thread is halted.

2.6. Gates

Gates provide protected control transfer, allowing a thread to jump to a predefined program counter in another address space with additional privilege. A gate object G has an ownership set O_G, a guard set G_G, and thread state, including the container entry of an address space, an initial program counter and stack pointer, and some closure arguments for the initial function. The guard set controls what other threads can invoke this gate, by requiring the caller to own all categories in G_G. A thread T can allocate a gate G only if $O_G \subseteq O_T$. A thread T' invoking G must specify a requested ownership set, O_R, to acquire upon invocation; invocation is permitted when $O_T \subseteq G_G$ and $O_R \subseteq (O_T \cup O_G)$. Gate objects are largely immutable (and thus subject to the parent container's label); the gate label L_G applies only to the gate object's (rarely-used) metadata.

Gates are often used like an RPC service. Unlike typical RPC, where the RPC server provides the resources to handle the request, gates allow the client to donate initial resources—namely, the thread object which invokes the gate. Gates can also be used to transfer privilege. The use of gates is discussed further in Section 3.5.

3. UNIX LIBRARY

Unix provides a general-purpose computing environment familiar to many people. In designing HiStar's user-level infrastructure, our goal was to provide as similar an environment to Unix as possible except in areas where there were compelling reasons not to—for instance, user authentication, which we redesigned for better security. As a result, porting software to HiStar is relatively straightforward; code that does not interact with security aspects such as user management often requires no modification.

HiStar's Unix environment is implemented in a library that emulates the Linux system call interface, comprising approximately 20,000 lines of code and providing abstractions like file descriptors, processes, fork and exec, file system, and signals. All of these abstractions are provided at user level, without any special privilege from the kernel. Thus, all information flow, such as obtaining the exit status of a child process, is made explicit in the Unix library. A vulnerability in the Unix library, such as a bug in the file system, compromises only threads that trigger the bug—an attacker can exercise only the privileges of the compromised thread, likely causing far less damage than a kernel vulnerability. An untrusted application, such as a virus scanner, can be isolated together with its Unix library, allowing for control over Unix vulnerabilities.

Most GNU software runs on HiStar without any source code modifications, including bash, gcc, gdb, and X; the main exception is OpenSSH, which requires small changes for user authentication and login code. The rest of this section discusses the design and implementation of our Unix emulation library.

3.1. File system

The HiStar file system uses segments and containers to implement files and directories, respectively. Each file corresponds to a segment object; to access the file contents, the segment is mapped into the thread's address space, and any reads or writes are translated into memory operations. The implementation coordinates with the user-mode page fault handler to return errors for invalid read or write requests. A file's length is defined to be the segment's length. Additional state, such as the modification time, is stored in the object's metadata.

A directory is a container with a special *directory segment* mapping file names to object IDs. A mutex in the directory segment serializes operations; for example, atomic rename within a directory is implemented by obtaining the directory's mutex lock, modifying the directory segment to reflect the new name, and releasing the lock.

Users that cannot write a directory cannot acquire the mutex, but they can still obtain a consistent view of directory segment entries by atomically reading a generation number and busy flag before and after reading each entry. The generation number is incremented by the library on each directory update.

Since file system objects correspond to HiStar kernel objects, permissions are specified in terms of labels, and are enforced by the kernel, not by the untrusted library file system code. For example, a file that should be accessible by only one user would be labeled $\{u_r, u_w\}$, where only that user owns u_r and u_w. A world-readable file that can be modified by only that user would be labeled $\{u_w\}$. Labels are similarly used for directories; read privilege on a directory allows listing the files in that directory and write privilege allows creating new files and renaming or deleting existing files.

3.2. Processes

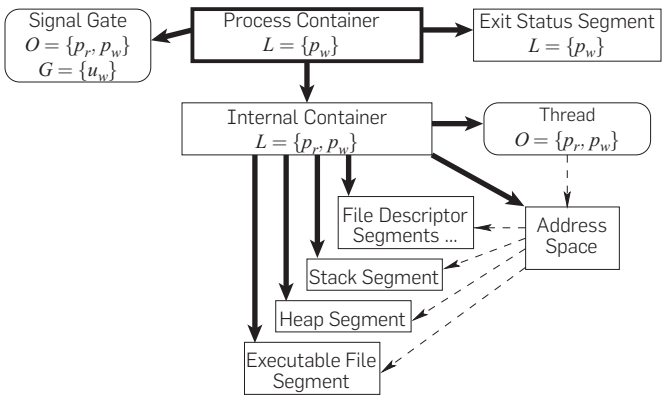
A process in HiStar is a user-space convention. Figure 4 illustrates the kernel objects that make up a typical process. Each process P has two categories, p_r and p_w, that protect its secrecy and integrity, respectively. Threads in a process typically own $\{p_r, p_w\}$, granting them full access to the process. The process consists of two containers: a process container and an internal container. The process container exposes objects that define the external interface to the process, such as a gate to receive signals from other processes (described in detail in the OSDI paper[18]) and a segment to store the process's exit status. The process container and exit status segment are labeled $\{p_w\}$, allowing other processes to read them, but not modify them (since other processes do not own this process's p_w). The internal container, address space, and segment objects are labeled $\{p_r, p_w\}$, preventing direct access by other processes.

3.3. File descriptors

All of the state typically associated with a file descriptor, such as the current seek position and open flags, is stored in a *file descriptor segment* in HiStar. Every file descriptor number corresponds to a specific virtual memory address. When a file descriptor is open in a process, the corresponding file descriptor segment is memory-mapped at the virtual address for that file descriptor number.

Typically each file descriptor segment has a label of $\{fd_r, fd_w\}$, where categories fd_r and fd_w grant read and write access to the file descriptor state. Access to the file descriptor is granted by granting ownership of $\{fd_r, fd_w\}$. Multiple processes can share file descriptors by mapping the same descriptor segment into their respective address spaces. By convention, every process adds hard links for all of its file descriptor segments to its own container. As a result, a shared descriptor segment is deallocated only when it has been closed and unlinked from the container of each process.

Figure 4. Structure of a HiStar process. A process container is represented by a thick border. Not shown are some label components that, e.g., ensure other users cannot read the exit status of this process. Bold and dashed lines represent hard and soft links.



3.4. Users

A pair of unique categories u_r and u_w define the read and write privileges of each Unix user U in HiStar, including root. Typically, threads running on behalf of user U own $\{u_r, u_w\}$, and a user's private files would have a label of $\{u_r, u_w\}$. One consequence of this design is that a single process can possess the privilege of multiple users, or perhaps multiple user roles, something that is hard to implement in Unix. On the other hand, our prototype does not support access control lists. (One way to implement access control lists would be to allocate a pair of categories for each ACL and to create a gate that would invoke code to evaluate the ACL rules and selectively grant ownership of these categories.) The authentication service, which verifies user passwords and grants user privileges, is described in more detail in Zeldovich et al.[18]

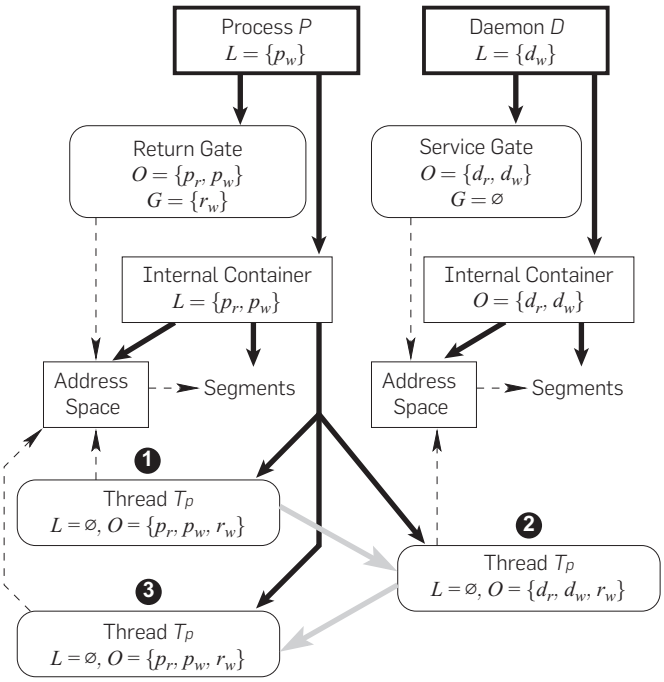
3.5. Gate calls

Gates provide a mechanism for implementing IPC. As an example, consider a phonebook service that allows looking up people's phone numbers by their name. Storing all names in a file may be undesirable, since users could easily obtain a list of all names. A HiStar process could provide this service by creating a *service gate* whose initial program counter corresponded to a function that looks up a name in a database that is accessible only to that process.

Gates in HiStar have no implicit return mechanism; the caller explicitly creates a *return gate* before invoking the service gate, which allows the calling thread to regain all of the privileges it had prior to calling the service. A *return category* r_w is allocated to prevent arbitrary threads from invoking the return gate; the return gate's guard set requires ownership of r_w to invoke the gate, and the caller grants ownership of r_w when invoking the service gate. Figure 5 illustrates a gate call from process P to daemon D.

While this design prevents a user from enumerating the daemon's private database, it does not ensure privacy of users' queries. If users do not trust the daemon to keep their queries private, they can enforce the privacy of their queries, as follows. The calling thread allocates a new secrecy category x_r and invokes the service gate in step 2 with a label of $\{x_r\}$ (instead of $\varnothing$). This thread can now read D's address space and any of D's segments, by virtue

Figure 5. Objects involved in a gate call operation. Thick borders represent process containers, bold lines represent hard links, and dashed lines represent soft links. r_w is the return category; d_r and d_w are the process read and write categories for daemon *D*. Three states of the same thread object T_p are shown: (1) just before calling the service gate, (2) after calling the service gate, and (3) after calling the return gate.



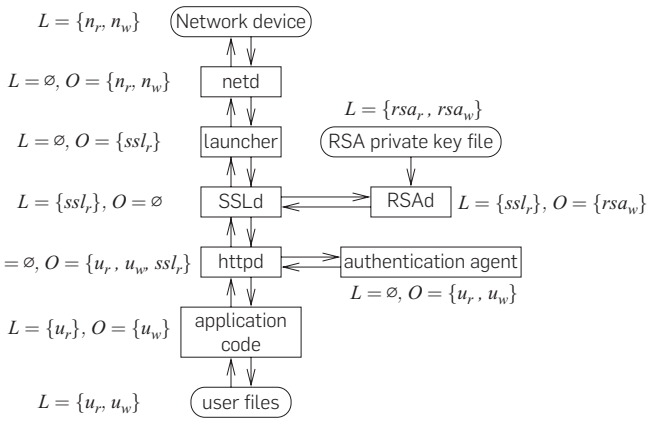
of owning d_r, but not modify them, since that would violate information flow constraints in category x_r. To continue executing, the thread makes a writable copy of the address space and its segments, labeled $\{d_r, d_w, x_r\}$. This effectively *forks D* to create a tainted clone. The thread can now read the database and return data to the caller, but cannot divulge any data to anyone that does not own x_r. Two considerations arise in this case. First, the tainted copy must be stored in some container. Since the thread is labeled x_r, the kernel does not allow it to store objects in *D*'s container (otherwise, the thread could leak information about the caller's private data to *D*) . Thus, before invoking the gate, the caller creates a container labeled x_r that can be used after invoking the gate. Second, when the thread returns to *P*'s address space through the return gate, its label still contains x_r. To remove x_r from the thread's label, *P* must have added x_r to the return gate's ownership label prior to invoking *D*.

Forking on tainted gate invocation is not appropriate for every service. Stateless services, such as a database lookup, are usually well-suited to forking, whereas services with mutable shared state may want to avoid forking by refusing tainted gate calls.

4. HiSTAR WEB SERVER

To illustrate how HiStar's protection mechanisms can be used by a real application, this section describes the HiStar

Figure 6. Architecture of the HiStar Web server. Rectangles represent processes. Rounded boxes represent devices and files. Arrows indicate communication, including gate calls. Not shown are process read and write categories from Figure 4.



SSL Web server. Figure 6 shows the server's overall architecture. The Web server is built from mutually distrustful components to reduce the effects of a compromise of any single component.

The TCP/IP stack in HiStar is implemented by a user-space process called *netd*, which has access to the kernel network device.[18] *netd* provides a traditional sockets interface to applications. Incoming TCP connections from Web browsers are initially accepted by the *launcher*. For each connection, the *launcher* spawns *SSLd*, to handle the SSL connection with the user's Web browser, and *httpd*, to process the user's plaintext HTTP request. The *launcher* then relays data between *SSLd* and the TCP connection. *SSLd*, in turn, uses the *RSAd* daemon to establish an SSL session key with the user's Web browser by generating an RSA signature using the SSL certificate private key kept by *RSAd*.

httpd receives the user's decrypted HTTP request from *SSLd* and extracts the user's password and the requested URL. It then authenticates the user by sending the user's password to that user's *password checking agent* from the HiStar authentication service.[18] If the authentication succeeds, *httpd* receives ownership of the user's secrecy and integrity categories, u_r and u_w, and executes the *application code* with the user's privileges (e.g., generating a PDF document). Application output is sent by *httpd* to the user's Web browser via *SSLd* for encryption.

4.1. Web server security

The HiStar Web server architecture has no hierarchy of privileges and no fully trusted components; instead, most components are mutually distrustful, and the effects of a compromise are typically limited to one user, usually the attacker himself. Figure 7 summarizes the security properties of this Web server, including the complexity of different components and effects of compromise.

The largest components in the Web server, *SSLd* and the application code, are minimally trusted and cannot disclose one user's private data to another user, even if

Figure 7. Components of the HiStar Web server, their complexity measured in lines of C code (not including libraries such as libc), their label and ownership, and the worst-case results of the component being compromised. The netd TCP/IP stack is a modified Linux kernel; HiStar also supports the lwIP TCP/IP stack, consisting of 35,000 lines of code, which has lower performance.

Component	Lines of Code	Label	Ownership	Effects of Compromise
netd	350,000	∅	$\{n_r, n_w\}$	Equivalent to an active network attacker; subject to same kernel label checks as any other process
launcher	310	∅	$\{ssl_r\}$	Obtain plaintext requests, including passwords, and subsequently corrupt user data
SSLd	340,000	$\{ssl_r\}$	∅	Corrupt request or response, or send unencrypted data to same user's browser
RSAd	4,600	$\{ssl_r\}$	$\{rsa_r\}$	Disclose the server's SSL certificate private key
httpd	300	∅	$\{u_r, u_w, ssl_r\}$	Full access to data in attacker's account, but not to other users' data
authentication	320	∅	$\{u_r, u_w\}$	Full access to data of the user whose agent is compromised, but no password disclosure
application	680,000+	$\{u_r\}$	$\{u_w\}$	Send garbage (only to the same user's browser), corrupt user data (for write requests)

they are malicious. The application code is confined by the user's secrecy category, u_r, and it is *httpd*'s job to ensure that the application code is labeled with u_r when *httpd* runs it. Although the application code owns the user's integrity category, u_w, this gives it the privilege to write only that one user's files, and not to export them. Ownership of u_w is necessary to allow the application code to read data not labeled with u_w. If the application code were to be labeled with u_w, it would be restricted to reading only data labeled u_w, which would exclude executables, shared libraries, and configuration files.

SSLd is confined by ssl_r, a fresh secrecy category allocated by the *launcher* for each new connection. Both the *launcher* and *httpd* own ssl_r, allowing them to freely handle encrypted and decrypted SSl data, respectively. However, *SSLd* can communicate only with *httpd* and, via the *launcher*, with the user's Web browser.

SSLd is not trusted to handle the SSL certificate private key. Instead, a separate and much smaller daemon, *RSAd*, has access to the private key and provides an interface to generate RSA signatures for SSL session key establishment. Not shown in Figure 7 is a category owned by *SSLd* that ensures no other process can invoke *RSAd*. Since *SSLd* is confined by ssl_r, the kernel ensures that *SSLd* cannot indirectly divulge any data to another process via its calls to *RSAd*, much as described in Section 3.5.

The HiStar authentication service used by *httpd* to authenticate users is described in detail in Zeldovich et al.,[18] but briefly, no code executes with all users' privileges, and the supplied password cannot be leaked even if the password checker is malicious. Our Web server does not use SSL client certificates for authentication. Doing so would require either trusting all of *SSLd* to authenticate users, or moving the client certificate code into the authentication agent. In comparison, the password checking agent is 320 lines of code.

One caveat of our prototype is its lack of SSL session caching. Because a separate instance of *SSLd* is used for each client request, clients cannot reuse existing session keys when connecting multiple times, requiring public key cryptography to establish a new session key. This limitation can be addressed by adding an SSL session cache that runs in a separate persistent process and owns all ssl_r categories, at the cost of increasing the amount of trusted code.

5. RELATED WORK

HiStar was directly inspired by Asbestos,[4] but differs in providing systemwide persistence and a lower-level kernel interface that closes known covert storage channels. While Asbestos is a message-passing system, HiStar relies heavily on shared memory. The HiStar kernel provides gates, not IPC, with the important distinction that upon crossing a gate, a thread's resources initially come from its previous domain. By contrast, Asbestos changes a process's label to track information flow when it receives IPCs, which is detectable by third parties and can leak information. Asbestos optimizes comparisons between enormous labels, which so far we have not done in HiStar.

Flume[8] showed how to provide information flow control on top of the Linux kernel, and introduced a cleaner label system (which HiStar and this paper have adopted). Flume also proposed *endpoints* to help programmers reason about labels on standard Unix abstractions; adopting endpoints in HiStar's Unix library would similarly help HiStar programmers. DStar[19] extended information flow control to decentralized systems and developed the HiStar Web server. Loki[20] showed how hardware can partially enforce HiStar's labels, to reduce the amount of fully trusted kernel code.

HiStar controls information flow with mandatory access control (MAC), a well-studied technique dating back decades.[1] The ADEPT-50 dynamically adjusted labels (essentially taint tracking) using the High-Water-Mark security model back in the late 1960s[9]; the idea has often resurfaced, for instance in IX[12] and LOMAC.[5] HiStar and its predecessor Asbestos are novel in that they make operations such as category allocation and untainting available to application programmers, where previous OSes reserved this functionality for security administrators. Decentralized untainting allows novel uses of categories that we believe promote better application structure and support applications such as Web services, which were not targeted by previous MAC systems.

Like HiStar, capability-based KeyKOS[3] and EROS[17] use a small number of kernel object types and a single-level store. HiStar's containers are reminiscent of hierarchical space banks in KeyKOS. However, while KeyKOS uses kernel-level capabilities to enforce labels at user level, HiStar bases all protection on kernel-level labels. The difference is significant because labels specify security properties while imposing less structure on applications; for example, an untrusted thread can dynamically alter its label to observe secret data, which has no analogue in a capability system.

The idea of using gates for protected control transfer

dates back to Multics.[16] Unlike Multics rings, HiStar's protection domains are not hierarchical. HiStar gates are more like doors in Spring.[6]

Decentralized untainting, while new in operating systems, was previously provided by programming languages, notably Jif.[13] Jif can track information flow at the level of individual variables and perform most label checks at compile time. However, Jif relies on the operating system for storage, trusted input files, administration, etc., which avoids many issues HiStar needs to address.

SELinux[11] lets Linux support MAC; like most MAC systems, policy is centrally specified by the administrator. In contrast, HiStar lets applications craft policies around their own categories. Retrofitting MAC to a large existing kernel such as Linux can be error-prone, especially given the sometimes ill-specified semantics of Linux system calls. HiStar's disciplined, small kernel can potentially achieve much higher assurance at the cost of compatibility.

6. DISCUSSION AND LIMITATIONS

The current prototype of HiStar supports x86-64, i386, SPARC, and ARM computers. The fully trusted kernel, including device drivers for any given machine, is approximately 20,000 lines of code. We expect that drivers can eventually be moved to untrusted user-space processes with the help of IOMMU hardware. We have found performance to be reasonable for Unix applications.[18]

Users familiar with Unix will find that, though HiStar resembles Unix, it also lacks several useful features and changes the semantics of some operations. For example, HiStar does not keep file access times; although possible to implement for some cases, tracking time of last access is in many situations fundamentally at odds with information flow control. Another difference is that *chmod*, *chown*, and *chgrp* revoke all open file descriptors and copy the file or directory. Because each file has one read and one write category, group permissions require a file's owner to be in the group. There is no file execute permission without read permission, and no setuid bit (though gates arguably provide a better alternative to both).

While trusted components can control how secret data is revealed, it is difficult to reason about *what* secret data is revealed. For example, *wrap* can ensure the scanner's output is sent only to the user's terminal, but it would be difficult to safely reveal even one bit of information from the scanner's output to the public (e.g., are any of the user's files infected?), since we must conservatively assume that the scanner's output may reveal *any* bit about the user's data.

7. SUMMARY

HiStar is a new operating system that provides strict information flow control without superuser privilege. Narrow interfaces allow for a small trusted kernel of less than 20,000 lines, on which a Unix-like environment is implemented in untrusted user-level library code. A new container abstraction lets administrators manage and revoke resources for processes they cannot observe. Side-by-side with the Unix environment, the system supports a number of high-security, privilege-separated applications previously not possible in a traditional Unix system. HiStar is available at http://www.scs.stanford.edu/histar/.

Acknowledgments

We thank Martin Abadi, Michael Reiter, and Michael Walfish for helping improve this paper, and many others that provided feedback on earlier papers.[18–20] This work was funded by joint NSF Cybertrust/DARPA grant CNS-0430425, by NSF Cybertrust award CNS-0716806, by the DARPA Application Communities (AC) program as part of the VERNIER project at Stanford and SRI International, and by a gift from Lightspeed Venture Partners. C

References

1. Bell, D.E., La Padula, L. *Secure Computer System: Unified Exposition and Multics Interpretation*. Technical Report MTR-2997, Rev. 1, MITRE Corporation, Bedford, MA, March 1976.
2. Biba, K.J. *Integrity Considerations for Secure Computer Systems*. Technical Report MTR-3153, MITRE Corporation, Bedford, MA, April 1977.
3. Bomberger, A.C., Frantz, A.P., Frantz, W.S., Hardy, A.C., Hardy, N., Landau, C.R., Shapiro, J.S. The KeyKOS nanokernel architecture. In *Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures*, April 1992, 95–112.
4. Efstathopoulos, P., Krohn, M., VanDeBogart, S., Frey, C., Ziegler, D., Kohler, E., Mazières, D., Kaashoek, F., Morris, R. Labels and event processes in the Asbestos operating system. In *Proceedings of the 20th SOSP* (Brighton, U.K., October 2005), 17–30.
5. Fraser, T. LOMAC: low water-mark integrity protection for COTS environments. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, CA, May 2000), 230–245.
6. Hamilton, G., Kougiouris, P. The Spring nucleus: a microkernel for objects. In *Proceedings of the Summer 1993 USENIX* (Cincinnati, OH, April 1993), 147–159.
7. Krohn, M., Efstathopoulos, P., Frey, C., Kaashoek, F., Kohler, E., Mazières, D., Morris, R., Osborne, M., VanDeBogart, S., Ziegler, D. Make least privilege a right (not a privilege). In *Proceedings of the 10th Workshop on Hot Topics in Operating Systems* (Santa Fe, NM, June 2005).
8. Krohn, M., Yip, A., Brodsky, M., Cliffer, N., Kaashoek, M.F., Kohler, E., Morris, R. Information flow control for standard OS abstractions. In *Proceedings of the 21st SOSP* (Stevenson, WA, October 2007), 321–334.
9. Landwehr, C.E. Formal models for computer security. *Comput. Surv.13,* 3 (September 1981), 247–278.
10. Leyden, J. Anti-virus vulnerabilities strike again. *The Register*, March 2005. http://www.theregister. co.uk/2005/03/18/mcafee_vuln/
11. Loscocco, P., Smalley, S. Integrating flexible support for security policies into the Linux operating system. In *Proceedings of the 2001 USENIX* (Boston, MA, June 2001), 29–40, FREENIX track.
12. McIlroy, M.D., Reeds, J.A. Multilevel security in the UNIX tradition. *Softw. Pract. Exp. 22*, 8 (1992), 673–694.
13. Myers, A.C., Liskov, B. Protecting privacy using the decentralized label model. *Trans. Comput. Syst. 9*, 4 (October 2000), 410–442.
14. Naraine, R. Symantec antivirus worm hole puts millions at risk. *eWeek.com*, May 2006. http://www.eweek. com/article2/0,1895,1967941,00.asp
15. Peterson, D. Anti-virus rife with vulnerabilities. *digitalbond.com*, January 2008. http://www.digitalbond.com/index.php/2008/01/07/anti-virus-rife-with-vulnerabilities/
16. Schroeder, M.D., Saltzer, J.H. A hardware architecture for implementing protection rings. In *Proceedings of the 3rd SOSP* (New York, March 1972), 42–54.
17. Shapiro, J.S., Smith, J.M., Farber, D.J. EROS: a fast capability system. In *Proceedings of the 17th SOSP* (Island Resort, SC, December 1999), 170–185.
18. Zeldovich, N., Boyd-Wickizer, S., Kohler, E., Mazières, D. Making information flow explicit in HiStar. In *Proceedings of the 7th OSDI* (Seattle, WA, November 2006), 263–278.
19. Zeldovich, N., Boyd-Wickizer, S., Mazières, D. Securing distributed systems with information flow control. In *Proceedings of the 5th NSDI* (San Francisco, CA, April 2008), 293–308.
20. Zeldovich, N., Kannan, H., Dalton, M., Kozyrakis, C. Hardware enforcement of application security policies. In *Proceedings of the 8th OSDI* (San Diego, CA, December 2008), 225–240.
21. Zoller, T. Clamav 0.94 and below—evasion and bypass due to malformed archive. April 2009. http://blog.zoller.lu/2009/04/clamav-094-and-below-evasion-and-bypass.html

Nickolai Zeldovich, Massachusetts Institute of Technology, CSAIL, Cambridge, MA.

Silas Boyd-Wickizer, Massachusetts Institute of Technology, CSAIL, Cambridge, MA.

Eddie Kohler, University of California, Los Angeles, CA.

David Mazières, Stanford University, Stanford, CA.

DOI:10.1145/2018396.2018420

Technical Perspective
A Perfect 'Match'

By William T. Freeman

WHAT MAKES AN array of pixel intensities look like a realistic image? How can you invent a set of plausible-looking image values in order to remove noise or to fill in missing regions of an image? These are problems the vision and image processing community has been struggling with for many years. Many different analytic approaches have been tried, but they seldom capture the richness and subtle details needed to produce realistic images.

To date, the best method to generate image data has been a surprisingly simple one: copy image values from somewhere else. This would seem to be too restrictive—how could one image patch possibly be a close enough match to another one to be useful? But it turns out that small image regions are essentially reusable parts, appearing, with small changes, many different times within an image or a set of images. If the patch is small enough, very good estimates of unknown image values can be found by extracting pixels from a patch with similar neighboring image values. The computer graphics community discovered this in the late 1990s and early 2000s, leading to an explosion of texture synthesis papers based on such sample-based methods.

This paradigm for finding good image values works not just for texture synthesis, but for many different image manipulations, too. The approach has been used for problems ranging from super-resolution to texture transfer, filling in, image editing, noise removal, and object detection. The source of image patches can be other regions of the image being processed, or other images. Sample-based image priors are ubiquitous in modern image processing and image synthesis.

Unfortunately, this powerful approach for image processing has a serious performance bottleneck: potentially for each pixel to be processed, one must find the database patch that is closest to the image data surrounding that pixel. With a naive approach, and searching for matches within the same image, the search cost can be quadratic in the number of image pixels.

Fortunately, it is seldom the case that the true nearest neighbor must be found; a patch that is very similar to the target patch is often all that's needed. This allows for existing fast, approximate nearest neighbor methods from the discrete algorithms community to be used. But while these methods help quite a bit, they often don't help enough, and we are still left with algorithms that may be too slow for interactive applications.

Which brings us to the breakthrough contributions in the paper that follows. The authors have developed an efficient way to find approximate nearest neighbors for the case of patches within image data.

Their advance resulted from two main insights. The first is the observation, also noted by others, that the best matches from two spatially neighboring positions are usually two spatially neighboring patches from the database region. This presents a fast method to guess a matching patch, given the match to the spatial neighbor, but such an approach can get stuck in solutions that are only locally optimal. The authors' fix to that comes from their second insight, delightfully counterintuitive: looking for a matching patch at *random* positions in the database region eventually finds good matches. Their "patch match" algorithm combines these approaches—deterministic update of a previous solution while allowing improvements from random guesses—to give a fast, approximate nearest neighbor algorithm for image patches that avoids getting stuck in bad solutions.

The authors have developed an efficient way to find approximate nearest neighbors for the case of patches within image data.

The breakthrough of the algorithm is its processing speed, which, for the first time, allows interactive use of some remarkable image editing algorithms that were previously restricted to slow, batch processing. The authors applied their algorithm to many image processing tasks, showing a broad range of applications.

The paper opens up algorithmic and theoretical questions. The scaling behavior with patch size is not known, nor is the best trade-off known for many of the choices made by the authors. But the work is having a large impact in the vision and graphics communities, both for the algorithm itself, and as an example of a class of algorithms to explore. It is unusual that commercial success follows so closely after an academic paper, but that happened in this case. The patch match algorithm is behind the release-defining "content-aware fill" feature of Adobe Photoshop CS5. C

William T. Freeman (billf@mit.edu) is a professor of computer science and Associate Department Head of the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology, Cambridge, MA.

VDOI:10.1145/2018396.2018421

The PatchMatch Randomized Matching Algorithm for Image Manipulation

By Connelly Barnes, Dan B Goldman, Eli Shechtman, and Adam Finkelstein

Abstract

This paper presents a new randomized algorithm for quickly finding approximate nearest neighbor matches between image patches. Our algorithm offers substantial performance improvements over the previous state of the art (20–100×), enabling its use in new interactive image editing tools, computer vision, and video applications. Previously, the cost of computing such matches for an entire image had eluded efforts to provide interactive performance. The key insight driving our algorithm is that the elements of our search domain—patches of image pixels—are correlated, and thus the search strategy takes advantage of these statistics. Our algorithm uses two principles: first, that good patch matches can be found via random sampling, and second, that natural coherence in the imagery allows us to propagate such matches quickly to surrounding areas. Our simple algorithm allows finding a single nearest neighbor match across translations only, whereas our general algorithm additionally allows matching of *k*-nearest neighbors, across all rotations and scales, and matching arbitrary descriptors. This one simple algorithm forms the basis for a variety of applications including image retargeting, completion, reshuffling, object detection, digital forgery detection, and video summarization.

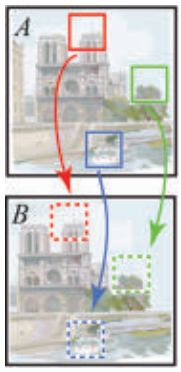
1. INTRODUCTION

As digital and computational photography have matured, researchers have developed sophisticated methods for analyzing and editing digital photographs and video. Many of the most powerful of these methods are *patch-based*: they divide the image into many small, overlapping rectangles of fixed size (e.g., 7×7 squares, one defined around every pixel), called patches, and then manipulate or analyze the image based on its patches. For example, patch-based techniques can be used for *image retargeting*, in which an image is resized to a new aspect ratio—the computer automatically produces a good likeness of the original image but with new dimensions. These techniques can also be used for *image completion*, in which a user simply erases an unwanted portion of an image, and the computer automatically synthesizes replacement pixels that plausibly match the rest of the image.

However, because these algorithms must search and manipulate millions of patches, performance in many cases had previously been far from interactive: operations such as image completion could previously take minutes.[24]

In this paper we describe an algorithm that accelerates many patch-based methods by at least an order of magnitude. This makes it possible to apply many powerful techniques for image editing for the first time in an interactive interface, as shown in Figure 1. We also offer intuitive controls for our image editing interface. Further, our algorithm is not limited to image editing, and can be applied to many techniques that use image patches. We report our experiences using our algorithm in object detection, digital forgery detection, and video summarization. Because our algorithm is a fairly general mathematical tool, we believe similar techniques could be used for other application domains in vision, graphics, or other fields where dense matchings are desired.

To understand our matching algorithm, we must consider the common components of patch-based algorithms: The core element of nonparametric patch sampling methods is a repeated search of all patches in one image region for the most similar patch in another image region. In other words, given images or regions A and B, find for every patch in A the nearest neighbor in B under a patch distance metric such as L_p. We call this mapping the *Nearest Neighbor Field* (NNF), illustrated schematically in the inset figure. Approaching this problem with a naïve brute force search is expensive—$O(mM^2)$ for image regions and patches of size M and m pixels, respectively. Even using acceleration methods such as *approximate nearest neighbors*[15] and dimensionality reduction, this search step remains the bottleneck of nonparametric patch sampling methods, preventing them from attaining interactive speeds. Furthermore, these tree-based acceleration structures use memory on the order of $O(M)$ or higher with relatively large constants, limiting their application for high resolution imagery.



To design an efficient search algorithm we look at the statistics of natural images—photographs of real objects—and design a good search strategy by taking

The original version of this paper is entitled "PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing" and was published in *ACM Transactions of Graphics (Proc. SIGGRAPH)*, August 2009.

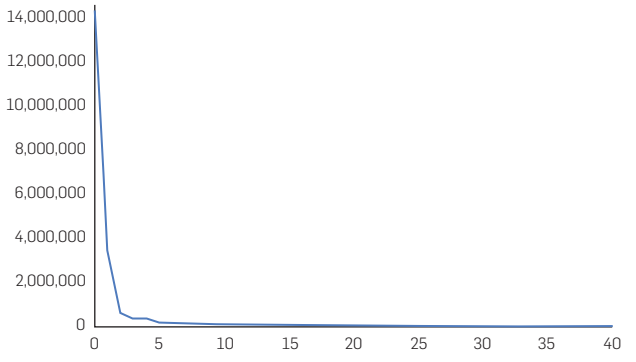
Figure 1. Manipulating images using our interactive tools. Left to right: (a) the original image; (b) a hole is marked (magenta) and we use line constraints (red/green/blue) to improve the continuity of the roofline; (c) the hole is filled in; (d) user-supplied line constraints for retargeting; (e) retargeting using constraints eliminates two columns automatically; and (f) user translates the roof upward using reshuffling.

(a) Original (b) Hole + constraints (c) Hole filled (d) Constraints (e) Constrained retarget (f) Reshuffle

Figure 2. Given an approximate match between patches with patch distance *D*, these 2D histograms show peaked distributions of the (*x*, *y*) coordinates where better matches are located. A better initial match with lower distance *D* causes more peaking. This is averaged over a dataset of more than 100 similar and dissimilar natural image pairs. Note the center pixel is black, but this is not visible at print resolution.

D = 50 *D* = 25 *D* = 10

Figure 3. Histogram showing a correlation between adjacent patches' nearest neighbors. On the horizontal axis, we measure how far apart are nearest neighbors of adjacent patches, in Euclidean 2D distance. On the vertical axis, we count the number of patches with a given distance. Most patches have zero Euclidean distance, indicating perfect coherence.



advantage of these statistics. Specifically, we look at the correlation between adjacent patches, and find that they are highly correlated. For example, as shown in Figure 2, given an approximate match between patches with patch distance D, the locations of matches with patch distance less than D are not uniformly distributed throughout the image, but instead follow a peaked distribution. We can also ask in the ground truth nearest neighbor matches, for two patches that are horizontally or vertically adjacent, how far apart spatially are their matches? This is visualized as a histogram in Figure 3 which again shows a peaked distribution.

Such biased distributions allow us to devise an efficient iterative search strategy for natural images—PatchMatch—that focuses computational effort on regions most likely to produce good matches. It converges for all images in the limit, but converges extremely quickly for natural images that follow its prior assumptions. We make several observations about the problem:

Dimensionality of offset space. First, although the dimensionality of the patch space is large (m dimensions), it is sparsely populated ($O(M)$ patches). Many previous methods have accelerated the nearest neighbor search by attacking the dimensionality of the patch space using tree structures (e.g., *kd*-tree), and dimensionality reduction methods (e.g., PCA). In contrast, our algorithm searches in the 2-D space of possible patch offsets, achieving greater speed and memory efficiency.

Natural structure of images. Second, the usual independent search for each pixel ignores the natural structure in images. In patch-sampling synthesis algorithms, the output typically contains large contiguous chunks of data from the input (as observed by Ashikhmin[1]). Thus we can improve efficiency by performing searches for adjacent pixels in an interdependent manner.

The law of large numbers. Finally, whereas any one random choice of patch assignment is very unlikely to be a good guess, some nontrivial fraction of a large field of random assignments will likely be good guesses. As this field grows larger, the chance that no patch will have a correct offset becomes vanishingly small.

Based on these three observations we offer a randomized algorithm for computing approximate NNFs using incremental updates (Section 3). The algorithm begins with an initial guess, which may be derived from prior information or may simply be a random field. The iterative process consists of two phases alternated at each patch: *propagation*, in which coherence is used to disseminate good solutions to adjacent pixels in the field; and *random search*, in which the current offset vector is perturbed by multiple scales of random offsets. Theoretical and empirical tests show the algorithm has good convergence properties for tested imagery up to 2 MP, and our CPU implementation shows speedups of 20–100 times versus *kd*-trees with PCA. Tree methods search in time $O(c_m mM)$, and incur the "curse of dimensionality:" c_m is exponential in the patch dimension m [15]. In contrast, our algorithm takes time $O(NmM)$, where N is the number of iterations (typically 5 when searching translations and 20 for rotations and scales). In

addition, unlike *kd*-trees, our algorithm uses substantially less auxiliary memory.

2. RELATED WORK

Patch-based sampling methods have become a popular tool for image and video synthesis and analysis. Applications include texture synthesis, image and video completion, summarization and retargeting, image recomposition and editing, image stitching and collages, new view synthesis, morphing, noise removal, super-resolution and more. We will next review some of these applications and discuss the common search techniques that they use as well as their degree of interactivity.

Nearest neighbor search methods. Correspondence searches can be classified as either *local*, where a search is performed in a limited spatial window, or *global*, where all possible displacements are considered. Correspondences can also be classified as *sparse*, determined only at a subset of key feature points, or *dense*, determined at every pixel or on a dense grid in the input. For efficiency, many algorithms only use local or sparse correspondences. Local search can only identify small displacements, so multiresolution refinement is often used (e.g., in optical-flow[3]), but large motions of small objects can be missed. Sparse keypoint[14] correspondences are commonly used for alignment, 3D reconstruction, and object detection and recognition. These methods work best on textured scenes at high resolution, but are less effective in other cases.

Dense patch-based methods. Those methods that find both dense and global matches have often had high time cost in the matching stage. Moreover, whereas in texture synthesis the texture example is usually a small image, in other applications such as patch-based completion, retargeting and reshuffling, the input image is typically much larger, making the search problem even more critical. Various speedups for this search have been proposed, generally involving tree structures such as TSVQ[23], *kd*-trees,[10, 24] and VP-trees,[12] each of which supports both exact and approximate search (ANN). The FLANN method[16] automatically chooses which tree algorithm to use according to the data. In synthesis applications, approximate search is often used in conjunction with dimensionality reduction techniques such as PCA,[10] because ANN methods are much more time- and memory-efficient in low dimensions. Ashikhmin[1] proposed a *local propagation* technique exploiting local coherence in the synthesis process by limiting the search space for a patch to the source locations of its neighbors in the exemplar texture. The propagation step of our algorithm is inspired by the same coherence assumption. The *k-coherence* technique[21] combines the propagation idea with a precomputation stage in which the *k* nearest neighbors of each patch are cached, and later searches take advantage of these precomputed sets. Although this accelerates the search phase, *k*-coherence still requires a full nearest-neighbor search for all pixels in the input. It assumes that the initial offsets are close enough that only a small number of nearest neighbors need to be searched. This may be true for small pure texture inputs, but we found that for large complex images our random search phase is required to escape local minima. In this work we compare speed and memory usage of our algorithm against *kd*-trees with dimensionality reduction, and we show that it is at least an order of magnitude faster than the best competing combination (ANN+PCA) and uses significantly less memory. Our algorithm also provides more generality than *kd*-trees because it can be applied with arbitrary nonlinear distance metrics, allows searching over additional continuous domains such as rotation and scale, and can be easily modified with constraints in the image space to preserve structures and enable local interactions. Locality sensitive hashing is an alternative to tree structures that can be used to search image patches,[18] but it also requires substantial additional memory and a precomputation step, unlike our algorithm.

Matching across rotations and scales. When search across a large range of scales and rotations is required, a dense search was previously considered impractical due to the high dimensionality of the search space. The common way to deal with this case is via keypoint detectors.[14] These detectors either find a local scale and orientation for each keypoint or do an affine normalization. These approaches are not always reliable due to image structure ambiguities and noise. Our generalized matching algorithm can operate on any common image descriptors (e.g., SIFT) and unlike many of the above tree structures, supports any distance function. Even while the algorithm naturally supports dense global matching, it may also be constrained to only accept matches in a local window if desired.

Texture synthesis and completion. Efros and Leung[9] introduced a simple nonparametric texture synthesis method that outperformed many previous model based methods by sampling patches from a texture example and pasting them in the synthesized image. Further improvements modify the search and sampling approaches for better structure preservation.[1, 23] The greedy fill-in order of these algorithms sometimes introduces inconsistencies when completing large holes with complex structures, but Wexler et al.[24] formulated a related problem of image completion as a global optimization, thus obtaining more globally consistent synthesis of large missing regions. This iterative multiscale optimization algorithm repeatedly searches for nearest neighbor patches for all hole pixels in parallel. Although their original implementation was typically slow (a few minutes for images smaller than 1 megapixel), our algorithm makes this technique applicable to much larger images at interactive rates. Patch optimization based approaches have now become common practice in texture synthesis.[22]

Control and interactivity. One advantage of patch sampling schemes is that they offer a great deal of fine-scale control. For example, in texture synthesis, the method of Ashikhmin[1] gives the user control over the process by initializing the output pixels with desired colors. The Image Analogies framework of Hertzmann et al.[10] uses auxiliary images as "guiding layers," enabling a variety of effects including super-resolution, texture transfer, artistic filters, and texture-by-numbers. In the field of image completion, impressive guided filling results were shown by annotating structures that cross both inside and outside the missing

region.[20] Lines are filled first using belief propagation, and then texture synthesis is applied for the other regions, but the overall run-time is on the order of minutes for a 0.5 megapixel image. Our system provides similar user annotations, for lines and other region constraints, but treats all regions in a unified iterative process at interactive rates.

Image retargeting. Many methods of image retargeting have applied warping or cropping, using some metric of saliency to avoid deforming important image regions.[25] Seam carving[2] uses a simple greedy approach to prioritize seams in an image that can safely be removed in retargeting. Although seam carving is fast, it does not preserve structures well, and offers only limited control over the results. Simakov et al.[19] proposed framing the problem of image and video retargeting as a maximization of bidirectional similarity between small patches in the original and output images, and a similar objective function and optimization algorithm was independently proposed by Wei et al.[22] as a method to create texture summaries for faster synthesis. Unfortunately, the approach of Simakov et al. is extremely slow compared to seam carving. Our constrained retargeting and image reshuffling applications employ the same objective function and iterative algorithm as Simakov et al., using our new nearest-neighbor algorithm to obtain interactive speeds.

Image "reshuffling" is the rearrangement of content in an image, according to user input, without precise mattes. Reshuffling was demonstrated simultaneously by Simakov et al.[19] and Cho et al.,[8] who used larger image patches and belief propagation in an MRF formulation. Reshuffling requires the minimization of a global error function, as objects may move significant distances, and greedy algorithms will introduce large artifacts. In contrast to all previous work, our reshuffling method is fully interactive. As this task might be particularly hard and badly constrained, these algorithms do not always produce the expected result. Therefore interactivity is essential, as it allows the user to preserve some semantically important structures from being reshuffled, and to quickly choose the best result among alternatives.

Object detection, digital forgeries, and collages. In addition to image editing, our algorithm can be applied to other problems in image analysis and video. For object detection, we take an approach similar to deformable template models.[11] Unlike previous approaches, we do not need to restrict our matching to sparse interest points or detect principle scales or orientations. We can instead use our matching algorithm to match all patches across all scales and orientations. For digital forgery detection, Popescu and Farid[17] previously demonstrated a method that can find regions of an image duplicated by a clone tool, by sorting image blocks after discarding JPEG compression artifacts. Our method works similarly. Our method is not robust to JPEG artifacts, but uses our more general matching algorithm, so it could potentially be generalized to find different types of forgeries such as those produced by our automatic hole filling. Finally, images can be stitched into collages[8, 19] using patch-based methods. We use this approach for our video summarization application.

3. MATCHING ALGORITHM

In this section we describe our core matching algorithm, which accelerates the problem of finding nearest neighbor patches by 20–100 x over previous work. Our algorithm is a randomized approximation algorithm: it does not always return the exact nearest neighbor, but returns a good approximate nearest neighbor quickly, and improves the estimate with each iteration.

For brevity's sake, in this paper we present a simplified version of the algorithm that searches for only one nearest neighbor per-patch, across only two translation dimensions. There will be more discussion of extensions later.

3.1. High level motivation

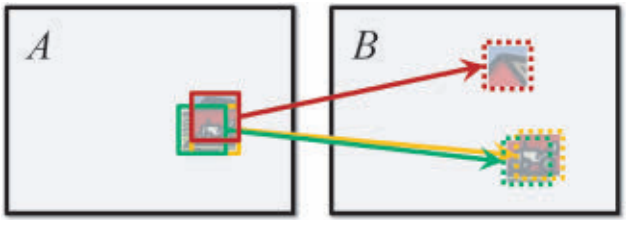
The high level intuition behind our algorithm is shown in Figure 4. We have two images A and B with patches visualized as colored rectangles. We wish to find for each patch in A the most similar patch in B. We do this by taking advantage of spatial locality properties: when we have a good match, we can *propagate* it to adjacent points on the image, and if we have a reasonable match, we can try to improve it by *randomly searching* for better matches around the target position.

We define a NNF as a function $\mathbf{f}: A \mapsto \mathbb{R}^2$ of nearest neighbors, defined over all possible patch coordinates (locations of patch centers) in image A, for some distance function of two patches D. Given patch coordinate $\mathbf{a}$ in image A and its corresponding nearest neighbor $\mathbf{b}$ in image B, $\mathbf{f}(\mathbf{a})$ is simply $\mathbf{b}$.[a] We refer to the values of $\mathbf{f}$ as *nearest neighbors*, and they are stored in an array whose dimensions are those of A.

As a reminder, our key insights are to search in the space of possible coordinate offsets, to search over adjacent patches cooperatively, and that even random coordinate assignments are likely to be a good guess for many patches over a large image.

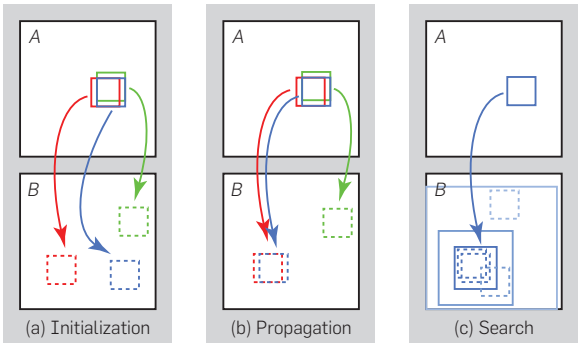
The algorithm has three main components, illustrated in Figure 5. Initially, the NNF is filled with either uniform random assignments or some prior information. Next, an iterative update process is applied to the NNF, in which good patch nearest neighbors are propagated to adjacent

Figure 4. Illustration of the matching problem. For each patch in image *A*, we find the most similar patch in image *B*. Sometimes patches that overlap have identical or coherent matches (shown in the yellow and green matches), and sometimes the matching coordinates are nearby (the red match).



[a] Our notation is in absolute coordinates, versus relative coordinates in Barnes et al.[6]

Figure 5. Phases of the randomized nearest neighbor algorithm: (a) patches initially have random assignments; (b) the blue patch checks above/green and left/red neighbors to see if they will improve the blue mapping, propagating good matches; (c) the patch searches randomly for improvements in concentric neighborhoods.



pixels, followed by random search in the neighborhood of the best nearest neighbor found so far. Sections 3.2 and 3.3 describe these steps in more detail.

3.2. Initialization

The NNF can be initialized either by assigning random values to the field, or by using prior information. When initializing with random values, we use independent uniform samples across the full range of image B. The image editing applications described in Section 4 repeat the search process in a coarse-to-fine pyramid, interleaving search and reconstruction at each scale. So we have the option to use the previous solution—possibly upscaled from a coarser level of the pyramid—as an initial guess. However, if we use only this initial guess, the algorithm can sometimes get trapped in suboptimal local minima. To retain the quality of this prior but still preserve some ability to escape from such minima, we perform a few early iterations of the algorithm using a random initialization, then merge with the initial guess only at patches where D is smaller, and then perform the remaining iterations. This gives a good tradeoff of retaining local minima found on previous iterations without getting stuck there.

3.3. Iteration

After initialization, we iteratively improve the NNF. Each iteration of the algorithm proceeds as follows: nearest neighbors are examined in scan order (from left to right, top to bottom), and each undergoes *propagation* followed by *random search*. These operations are interleaved at the patch level: if P_j and S_j denote, respectively, propagation and random search at patch j, then we proceed in the order: P_1, S_1, P_2, S_2,..., P_n, S_n.

Propagation. We attempt to improve $\mathbf{f}(x,y)$ using the known nearest neighbors of $\mathbf{f}(x-1, y)$ and $\mathbf{f}(x, y-1)$, assuming that the patch coordinates are likely to be offset by the same relative translation one pixel to the right or down. For example, if there is a good mapping at $(x-1, y)$, we try to use the translation of that mapping one pixel to the right for our mapping at (x, y). Let $\mathbf{z} = (x, y)$. The new candidates for $\mathbf{f}(\mathbf{z})$ are $\mathbf{f}(\mathbf{z} - \Delta_p) + \Delta_p$, where Δ_p takes on the values of (1, 0) and (0, 1). Propagation takes a downhill step if either candidate provides a smaller patch distance D.

The effect is that if (x, y) has a correct mapping and is in a coherent region R, then all of R below and to the right of (x, y) will be filled with the correct mapping. Moreover, on *even* iterations we propagate information *up and left* by examining patches in reverse scan order, and using candidates below and to the right. If used in isolation, propagation converges very quickly, but ends up in a local minimum. So a second set of trials employs *random search*.

Random search. A sequence of candidates is sampled from an exponential distribution, and the current nearest neighbor is improved if any of the candidates has smaller distance D. Let $\mathbf{v}_0$ be the current nearest neighbor $\mathbf{f}(\mathbf{z})$. We attempt to improve $\mathbf{f}(\mathbf{z})$ by testing a sequence of candidate mappings at an exponentially decreasing distance from $\mathbf{v}_0$:

$$\mathbf{u}_i = \mathbf{v}_0 + w\alpha^i \mathbf{R}_i \quad (1)$$

where $\mathbf{R}_i$ is a uniform random in $[-1, 1] \times [-1, 1]$, w is a large maximum search "radius," and α is a fixed ratio between search window sizes. We examine patches for $i = 0, 1, 2,...$ until the current search radius $w\alpha^i$ is below 1 pixel. In our applications w is the maximum image dimension, and $\alpha = 1/2$, except where noted. Note the search window must be clamped to the bounds of B.

Halting criteria. Although different criteria for halting may be used depending on the application, in practice we have found it works well to iterate a fixed number of times. All the results shown here were computed with 4–5 iterations total, after which the NNF has almost always converged. Convergence is illustrated in Figure 6.

Efficiency. The efficiency of this naive approach can be improved in a few ways. In the propagation and random search phases, when attempting to improve an offset $\mathbf{f}(\mathbf{z})$ with a candidate offset $\mathbf{u}$, *early termination* can be used if a partial sum for the patch distance exceeds the current best known patch distance. Also, in the propagation stage, when using square patches of side length p and an L_q norm, the change in distance can be computed incrementally in $O(p)$ rather than $O(p^2)$ time, by noting redundant terms in the summation over the overlap region. However, this incurs additional memory overhead to store the current best distances $D(\mathbf{f}(x, y))$.

3.4. Discussion

Our algorithm converges quickly to a good approximate solution in a small number of iterations. We compared our convergence with competing methods such as *kd*-trees with PCA, vp-trees with PCA, and theoretically analyzed the convergence properties of our algorithm.[6] We found that for equal matching error, and low numbers of iterations, our algorithm is 20–100 x faster than the best competing algorithm, *kd*-tree with PCA, and uses substantially less memory.

Figure 6. Illustration of convergence. (a) The top image is reconstructed using only patches from the bottom image, (b) above: the reconstruction by patch "voting" (each patch looks up its nearest neighbor's colors, and these are averaged for all overlapping patches), below: a random initial offset field, with magnitude visualized as saturation and angle visualized as hue, (c) 1/4 of the way through the first iteration, high-quality offsets have been propagated in the region above the current scan line (denoted with the horizontal bar). (d) 3/4 of the way through the first iteration, (e) first iteration complete, (f) two iterations, and (g) after five iterations, almost all patches have stopped changing.

(a) Originals (b) Random (c) $\frac{1}{4}$ iter (d) $\frac{3}{4}$ iter (e) 1 iter (f) 2 iters (g) 5 iters

Our algorithm bears some superficial similarity to Belief Propagation and Graph Cuts algorithms often used to solve Markov Random Fields on an image grid. However, unlike the MRF models, our algorithm does not have a neighborhood term that explicitly creates smooth or coherent matches. Because our search algorithm finds coherent regions in early iterations, our matches implicitly err towards coherence. Thus our approach is sufficient for many practical synthesis applications, while avoiding the computational expense of MRF approaches.

4. APPLICATIONS

We have presented a fast algorithm for finding good matches between patches in arbitrary images. Based on our algorithm, we have developed an interactive interface for editing images, using sophisticated patch-based synthesis techniques. Our synthesis uses the framework of Simakov et al.,[19] which was previously demonstrated on synthesis tasks such as image collages, reshuffling, retargeting, automatic cropping, and the analogues of these in video. This method works by iteratively improving a current output image, starting at a coarse resolution and proceeding to finer resolutions, in each iteration repeatedly making sure all patches in the source image are present in the current result, and vice versa. Thus at the core of this method is our correspondence algorithm, which is used to query patches in both query directions. If instead we define a region of undesired content (a "hole") to be removed from an image, the method of Simakov becomes very similar to the image completion algorithm of Wexler et al.[24] Thus, undesired objects can be removed from photographs using the same synthesis framework. However, Simakov et al. reported several minutes to generate output images. Because our object removal technique offers high quality results with low synthesis time, it is suitable for commercial deployment, and has been implemented as the new Content-Aware Fill feature in Adobe Photoshop CS5.

For many of these applications, our interactive interface allows the user to receive feedback in seconds. Many synthesis techniques are therefore interactive for the first time because of our algorithm. We also offer the user new interactive controls for guiding the output image with constraints.

Figure 7. Example of guided image completion. The bird is removed from input (a). The user marks the completion region and labels constraints on the search in (b), producing the output (c) in a few seconds.

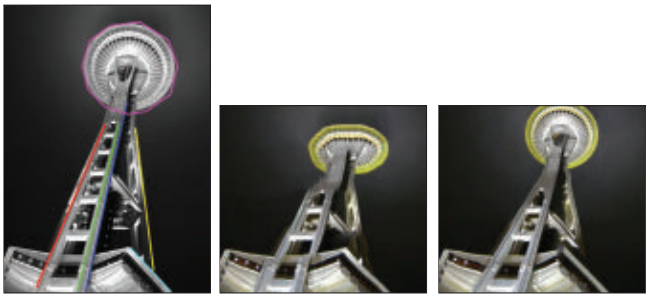
(a) Input (b) Hole and guides (c) Completion result

We show a number of example user interactions in Figure 1. These include automatically replacing an undesired image region, as well as retargeting the image to change aspect ratio (using constraints to prevent pillars from breaking), and reshuffling, or moving up the roof of the building. In Figure 7 we show removal of an undesired object. In Figure 8 we show how some of our constraints can prevent a building from bending or breaking during the retargeting process. Because our synthesis framework works by iteratively modifying the output image to the desired resolution, features tend to bend or break slowly, so constraints can be applied during the iterative process to prevent breaks. In Figure 9 we show a local scale tool that allows a region to be scaled while preserving texture. In Figure 10 we show how many of these tools can be combined in a workflow of editing architecture.

We have also investigated additional applications such as object detection, label transfer, symmetry detection, denoising, detecting digital forgeries,[7] and video summarization.[5] We present some results for object detection, forgery detection, and video summarization here.

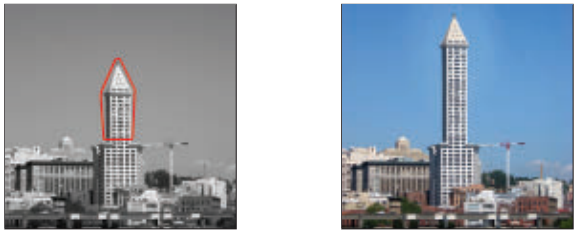
For many of these applications, more general variants of our matching algorithm are needed. We have developed a generalization of our matching algorithm[7] that searches over all rotations and scales (useful for object detection), finds k nearest neighbors instead of a single nearest neighbor[b] (useful for detecting digital forgeries), and matches

Figure 8. Constraints. The original image (a) is retargeted without constraints (b). Constraints indicated by colored lines produce straight lines and the circle is scaled down to fit the limited space (c).



(a) Original (b) Retargeted (c) With constraints

Figure 9. Example using local scale tool. The user marks a source polygon (a), and then applies a nonuniform scale (b) to the polygon, while preserving texture.



(a) Building marked by user (b) Scaled up, preserving texture

arbitrary descriptors, that is, vectors computed at each pixel (useful for matching features, e.g., SIFT features that are robust to camera and lighting changes). These generalizations are simple and natural extensions of our original algorithm.

In Figure 11 we show an example of object detection. The algorithm accepts a template image to be found within a large target image. The template is then located by breaking both images into small square patches, then running our matching algorithm across all rotations and scales, with a patch descriptor that compensates for changes in lighting. Then for each template object, the resulting NNF is used to estimate the pose, after rejecting outliers due to occlusions and poor matches.

We show a second application of detecting digital forgeries made by the "clone brush" in Figure 12. When a user forges an image in this way, he or she removes an object by manually replacing it with a different region of the same image. Therefore, in the forged image some patches are duplicated in large coherent regions. We detect these by using our matching algorithm to find, for each patch, its *k*-nearest neighbors within the same image. Then we detect cloned regions by locating large regions in the NNF that are roughly coherent.

We finally present our video summarization system,[5] shown in Figure 13. This system automatically selects and collages

[b] Note that Liu and Freeman[13] also investigated *k*-NN search based on our algorithm.

Figure 10. Modifying architecture with reshuffling. The images contain many repetitions, so the algorithm can often produce plausible output even when subject to extreme constraints.

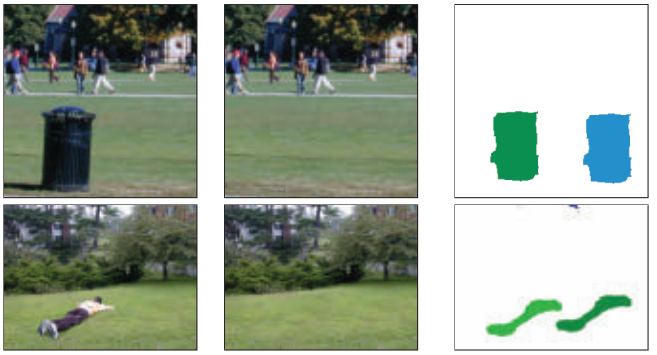
Figure 11. Detecting objects. Templates, left, are matched to the image, right. Square patches are matched, searching over all rotations and scales.

video frames to produce a seamlessly zoomable visual timeline. This timeline can be used as an alternative to the simple scrollbars typically used in video players. The user can select a desired scene to move to with the mouse. Or, to see more details from a given part of the film, the user can smoothly zoom in to expose more details from that part of the film. We produce our collages using patch-based synthesis, and because of the speed of our algorithm, we can produce timelines interactively.

5. FUTURE WORK

We believe our algorithm can be extended to different search domains such as 1D (e.g., audio) and 3D geometry, and allow

Figure 12. Detecting image regions forged using the clone brush: (a) the original, untampered image, (b) the forged image, and (c) the cloned regions detected by our algorithm. (Imagery courtesy of Popescu and Farid.[17])



(a) Original (b) Forged (c) Detected forgery

Figure 13. A multiscale tapestry represents an input video as a seamless and zoomable summary image that can be used to navigate through the video. This visualization eliminates hard borders between frames, providing spatial continuity and also continuous zooms to finer temporal resolutions. This figure depicts three discrete scale levels for the film *Elephants Dream* (Courtesy of the Blender Foundation). The lines between the scale levels indicate the corresponding domains between scales.

for other new applications such as synthesis of 3D geometry or stereo depth maps. For extensive detail on the matching algorithms, image statistics, applications, and directions for future work, consult Barnes.[4]

Our research has led us to an important conclusion about the design of image manipulation algorithms: by understanding the natural statistics of a problem domain, one can often customize a solution strategy around those statistics. In our case, by understanding the correlations between different nodes (pixels of an image), we designed the search strategy to take advantage of these statistics. We are excited about the potential of our techniques to accelerate search in many different domains, as well as the future work it has opened up.

Acknowledgments

We would like to thank the following Flickr users for Creative Commons imagery: *Sevenbrane* (Greek temple), *Wili* (boys), *Moi of Ra* (flowers), and *Celie* (pagoda). This work was sponsored in part by Adobe Systems and the NSF grant IIS-0511965. C

References

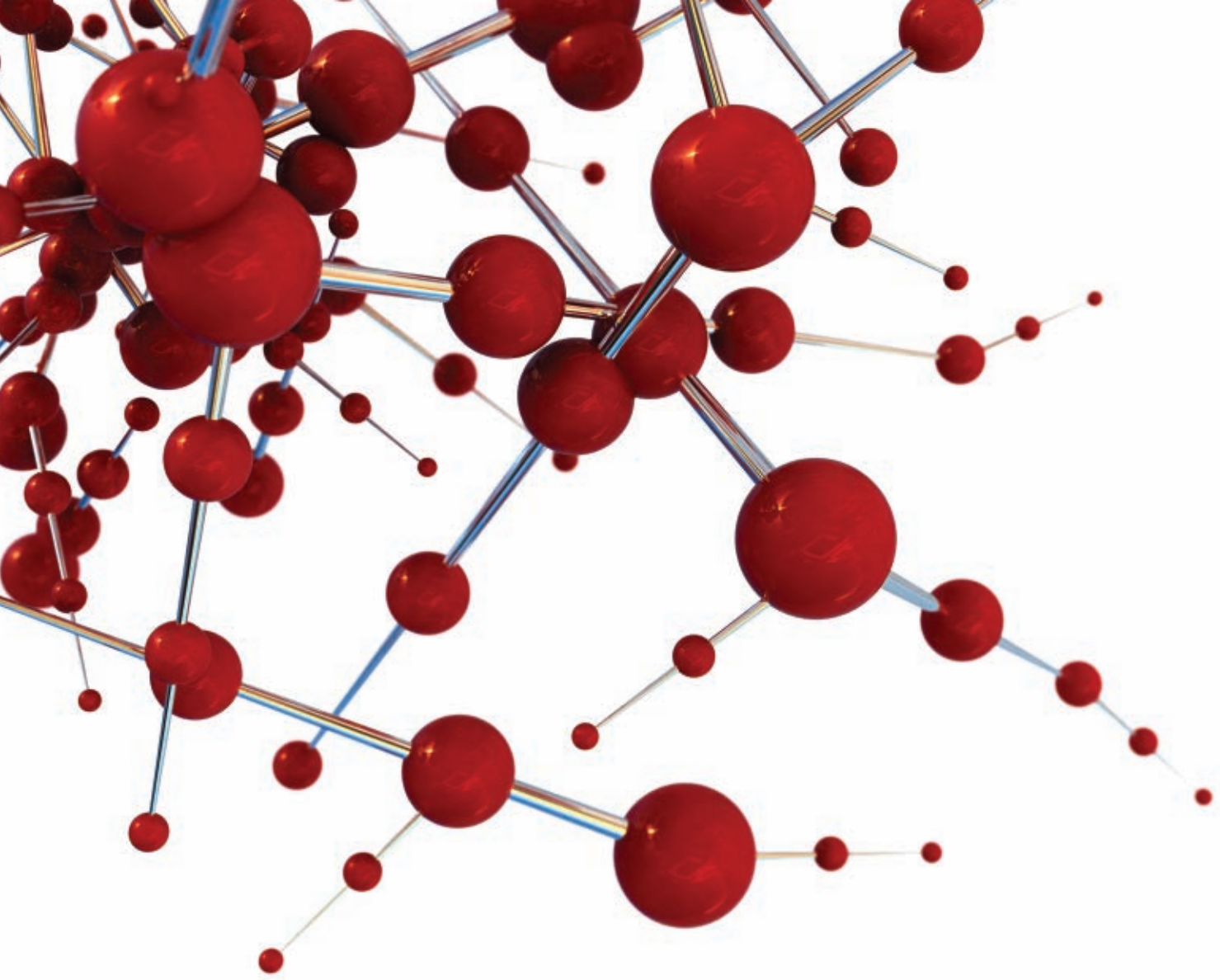
1. Ashikhmin, M. Synthesizing natural textures. In *I3D Proceedings* (2001). ACM, 217–226.
2. Avidan, S., Shamir, A. Seam carving for content-aware image resizing. *ACM Trans. Gr. (Proc. SIGGRAPH) 26*, 3 (2007), 10.
3. Baker, S., Scharstein, D., Lewis, J., Roth, S., Black, M., Szeliski, R. A database and evaluation methodology for optical flow. In *Proceedings of ICCV*, volume 5, 2007.
4. Barnes, C. PatchMatch: A Fast Randomized Matching Algorithm with Application to Image and Video. Ph.D. thesis. Princeton University, Princeton, NJ. May 2011.
5. Barnes, C., Goldman, D.B., Shechtman, E., Finkelstein, A. Video tapestries with continuous temporal zoom. *ACM Trans. Gr. (Proc. SIGGRAPH) 29*, 3 (Aug. 2010).
6. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Gr. (Proc. SIGGRAPH) 28*, 3 (2009), 24.
7. Barnes, C., Shechtman, E., Goldman, D.B., Finkelstein, A. The generalized PatchMatch correspondence algorithm. In *ECCV*, Sept. 2010.
8. Cho, T.S., Butman, M., Avidan, S., Freeman, W. The patch transform and its applications to image editing. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
9. Efros, A.A., Leung, T.K. Texture synthesis by non-parametric sampling. *IEEE ICCV 2* (1999), 1033.
10. Hertzmann, A., Jacobs, C.E., Oliver, N., Curless, B., Salesin, D. Image analogies. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2001), 327–340.
11. Jain, A., Zhong, Y., Dubuisson-Jolly, M. Deformable template models: A review. *Signal Process. 71*, 2 (1998), 109–129.
12. Kumar, N., Zhang, L., Nayar, S.K. What is a good nearest neighbors algorithm for finding similar patches in images? In *ECCV* (2008), II, 364–378.
13. Liu, C., Freeman, W. A high-quality video denoising algorithm based on reliable motion estimation. In *Proceedings of the ECCV*, 2010.
14. Mikolajczyk, K., Schmid, C. A performance evaluation of local descriptors. *IEEE Pattern Anal. Mach. Intell. (PAMI) 27*, 10 (2005), 1615–1630.
15. Mount, D.M., Arya, S. ANN: A library for approximate nearest neighbor searching. Oct. 28, 1997.
16. Muja, M., Lowe, D. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009.
17. Popescu, A., Farid, H. *Exposing Digital Forgeries by Detecting Duplicated Image Regions*. Technical Report. Department of Computer Science, Dartmouth College, 2004.
18. Shapira, L., Avidan, S., Shamir, A. Mode-detection via median-shift. In *IEEE Computer Vision and Pattern Recognition (CVPR)* (2009), IEEE, 1909–1916.
19. Simakov, D., Caspi, Y., Shechtman, E., Irani, M. Summarizing visual data using bidirectional similarity. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, Anchorage, AK, 2008.
20. Sun, J., Yuan, L., Jia, J., Shum, H.-Y. Image completion with structure propagation. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2005), 861–868.
21. Tong, X., Zhang, J., Liu, L., Wang, X., Guo, B., Shum, H.-Y. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Gr. (Proc. SIGGRAPH) 21*, 3 (July 2002), 665–672.
22. Wei, L.-Y., Han, J., Zhou, K., Bao, H., Guo, B., Shum, H.-Y. Inverse texture synthesis. *ACM Trans. Gr. (Proc. SIGGRAPH) 27*, 3 (2008).
23. Wei, L.Y., Levoy, M. Fast texture synthesis using tree-structured vector quantization. In *ACM Transactions on Graphics (Proc. SIGGRAPH)* (2000), 479–488.
24. Wexler, Y., Shechtman, E., Irani, M. Space-time completion of video. *IEEE Pattern Anal. Mach. Intell. (PAMI) 29*, 3 (2007), 463–476.
25. Wolf, L., Guttmann, M., Cohen-Or, D. Non-homogeneous content-driven video-retargeting. In *IEEE ICCV*, 2007.

Connelly Barnes, Princeton University, Princeton, NJ.

Dan B Goldman, Adobe Systems, Seattle, WA.

Eli Shechtman, Adobe Systems, Seattle, WA.

Adam Finkelstein, Princeton University, Princeton, NJ.



CAREERS

American Association for the Advancement of Science

AAAS Science & Technology Policy Fellowships

Apply your science to serve society!

Since 1973, more than 2,200 scientists and engineers have contributed their analytical skills to policymaking in Washington, DC, while learning about the role of science in the federal government.

Career-enhancing opportunities are available in more than 30 Congressional offices and 15 federal agencies for science and engineering professionals at all career stages.

Application Deadline: December 5

Visit http://fellowships.aaas.org for more details.

Applicants must be US citizens and hold a doctoral level degree (PhD, MD, DVM, etc.) in any scientific discipline, or a master's degree in engineering with three years of post-degree experience.

Baylor University

Assistant, Associate or Full Professor of Computer Science

The Department of Computer Science seeks a productive scholar and dedicated teacher for a tenure-track position beginning August, 2012. The ideal candidate will hold a terminal degree in Computer Science or closely related field, demonstrate scholarly capability and an established and active independent research agenda in one of several core areas of interest, including, but not limited to, game design and development, software engineering, computational biology, machine learning or large-scale data mining. A successful candidate will also exhibit a passion for teaching and mentoring at the graduate and undergraduate level. For position details and application information please visit: http://www.baylor.edu/hr/index.php?id=81302

The Department: The Department offers a CS-AB-accredited B.S. in Computer Science degree, a B.A. degree with a major in Computer Science, a B.S. in Informatics with a major in Bioinformatics, and a M.S. degree in Computer Science. The Department has 13 full-time faculty members, over 250 undergraduate majors and approximately 30 master's students. We are currently seeking approval to offer a dual Ph.D. degree in cooperation with a well-established partner institution. Interested candidates may contact any faculty member to ask questions and/or visit the web site of the School of Engineering and Computer Science at http://www.ecs.baylor.edu.

The University: Chartered in 1845 by the Republic of Texas, Baylor University is the oldest university in Texas and the world's largest Baptist University. It is situated on a 500-acre campus next to the Brazos River and annually enrolls more than 14,000 students in over 150 baccalaureate and 80 graduate programs. Baylor's mission is to educate men and women for worldwide leadership and service by integrating academic excellence and Christian commitment within a caring community. Baylor is actively recruiting new faculty with a strong commitment to the classroom and an equally strong commitment to discovering new knowledge as Baylor aspires to become a top tier research university while reaffirming and strengthening its distinctive Christian mission as described in Baylor 2012 (www.baylor.edu/vision/).

Application Procedure: Applications, including detailed curriculum vitae, a statement demonstrating an active Christian faith, and contact information for three references should be sent to: Chair Search Committee, Department of Computer Science, Baylor University, One Bear Place #97356, Waco, TX 76798-7356.

Appointment Date: Fall 2012. For full consideration, applications should be received by January 1, 2012.

Baylor is a Baptist university affiliated with the Baptist General Convention of Texas. As an Affirmative Action/Equal Employment Opportunity employer, Baylor encourages minorities, women, veterans, and persons with disabilities to apply.

Baylor University

Assistant or Associate Professor of Computer Science

Chartered in 1845 by the Republic of Texas, Baylor University is the oldest university in Texas and the world's largest Baptist University. Baylor's mission is to educate men and women for worldwide leadership and service by integrating academic excellence and Christian commitment within a caring community. Baylor is actively recruiting new faculty with a strong commitment to the classroom and an equally strong commitment to discovering new knowledge as Baylor aspires to become a top tier research university while reaffirming and strengthening its distinctive Christian mission as described in Baylor 2012 (www.baylor.edu/vision/). The combination of teaching, research and service has made Baylor one of the best universities for faculty, according to the Chronicle of Higher Education http://chronicle.com/article/Great-Colleges-to-Work-For/128312/.

The Department of Computer Science seeks a productive scholar and dedicated teacher for a tenure-track position beginning August, 2012. All specializations will be considered. Game/simulated environments, mobile computing, and graphics are of particular interest. The successful candidate will hold a terminal degree in Computer Science or a closely related field, demonstrate scholarly capability in his or her area of specialization, and exhibit a passion for teaching and mentoring at the graduate and undergraduate level. For position details and application information please visit: http://www.ecs.baylor.edu.

The Department: The Department offers a CS-AB-accredited B.S. in Computer Science degree, a B.A. degree with a major in Computer Science, a B.S. in Informatics with a major in Bioinformatics, and a M.S. degree in Computer Science. We are currently seeking approval to offer a dual Ph.D. degree in cooperation with a well-established European institution. The Department has 15 full-time faculty, over 370 undergraduate majors and 30 master's students. The Department's greatest strength is the faculty's dedication to the success of the students and each other. Interested candidates may contact any faculty member to ask questions and/or visit the web site of the School of Engineering and Computer Science at http://www.ecs.baylor.edu

The University: Baylor University, situated on a 500-acre campus next to the Brazos River. It annually enrolls more than 14,000 students in over 150 baccalaureate and 80 graduate programs through: the College of Arts and Sciences; the Schools of Business, Education, Engineering and Computer Science, Music, Nursing, Law, Social Work, and Graduate Studies; plus Truett Seminary and the Honors College. For more information see http://www.baylor.edu.

Application Procedure: Please submit a letter of application, current curriculum vitae, and transcripts. Include names, addresses, and phone numbers of three individuals from whom you have requested letters of recommendation to: Jeff Donahoo, Ph.D., Search Committee Chair, Baylor University, One Bear Place #97356, Waco, Texas 76798-7356, Materials may be submitted to: Jeff_Donahoo@baylor.edu

Appointment Date: Fall 2012. For full consideration, applications should be received by January 1, 2012. However, applications will be accepted until the position is filled.

Baylor is a Baptist university affiliated with the Baptist General Convention of Texas. As an Affirmative Action/Equal Employment Opportunity employer, Baylor encourages minorities, women, veterans, and persons with disabilities to apply.

Boston University

Department of Electrical & Computer Engineering (ECE)

Faculty Positions in Computer Engineering

The Department of Electrical & Computer Engineering (ECE) at Boston University (BU) is seeking candidates for anticipated faculty positions in Computer Engineering. All areas and ranks will be considered, with particular interest in entry-level candidates in software, security, and computer systems. The Department is seeking to foster growth in the broad, interdisciplinary topics of energy, health, information systems, and

cyberphysical systems. Candidates with research interests that transcend the traditional boundaries of ECE are strongly encouraged to apply. Joint appointments with other BU departments and with the Division of Material Science & Engineering and Division of Systems Engineering are possible for candidates with appropriate experience and interests.

Qualified candidates must possess a relevant, earned PhD, and have a demonstrable ability to teach effectively, develop funded research programs in their area of expertise, and contribute to the tradition of excellence in research that is characteristic of the ECE Department. Self-motivated individuals who thrive on challenge and are eager to utilize their expertise to strengthen an ambitious program of departmental enhancement are desired. Women, minorities, and candidates from other underrepresented groups are especially encouraged to apply and help us continue building an exceptional 21st century university department.

ECE at BU is a world-class department with excellent resources that is steadily gaining national and international prominence for its exceptional research and education record. ECE is part of BU's rapidly growing and innovative College of Engineering, and currently consists of 40 faculty members, 200 graduate students, and 250 BS majors. Outstanding collaboration opportunities are available with nationally recognized medical centers and universities/colleges, nearby research centers, and industry throughout the Boston area.

Beyond its research and academic activities, BU has a lively, urban campus situated along the banks of the Charles River in Boston's historic Fenway-Kenmore neighborhood. The campus and surrounding areas offer limitless opportunities for recreational activities, from world-class art and performances to sporting events and fine dining.

Please visit http://www.bu.edu/ece/facultysearch for instructions on how to apply. Application deadline is December 31, 2011. The review of applications will begin on October 1, 2011. Therefore, applicants are encouraged to apply early. Boston University is an Equal Opportunity/Affirmative Action Employer.

Carleton College
Assistant Professor of Computer Science

Carleton College invites applications for a one-year position potentially renewable for a second year) in computer science, in any area of specialization, beginning September 1, 2012.

Carleton is a highly selective liberal arts college with outstanding, enthusiastic students. We seek an equally enthusiastic computer scientist committed to excellence in teaching, curriculum design, ongoing research, and undergraduate research advising. We are particularly interested in applicants who will strengthen the departmental commitment to students from underrepresented groups. To learn more about the position or to apply, visit jobs.carleton.edu. Applications completed by December 16, 2011 will receive full consideration.

Carleton College does not discriminate in providing employment. Please view the description for this position at jobs.carleton.edu for Carleton's full anti-discrimination statement.

Cornell University
Multiple Faculty Positions

Multiple faculty positions are available at Cornell's Department of Computer Science. Candidates are invited to apply at all levels including tenured, tenure-track, or lecturer. We are interested in applications from any area of computer science, including artificial intelligence, computational biology, cryptography, databases, game design, graphics, machine learning, networking, programming languages, robotics, security, scientific computing, systems, and theory of computation.

Applicants for tenured and tenure track must hold a Ph.D. and have demonstrated an ability to conduct outstanding research. Lecturers must hold the equivalent of a Master's degree, with a Ph.D. preferred. To ensure full consideration, applications should be received by December 1, 2011, but will be accepted until all positions are filled. Applicants should submit a curriculum vita, brief statements of research and teaching interests, and arrange to have at least three references letters submitted at http://www.cs.cornell.edu/apply.

Cornell University is an equal opportunity, affirmative action educator and employer; qualified women and minority candidates are particularly encouraged to apply.

Duke University
Department of Computer Science

The Department of Computer Science at Duke University in Durham, North Carolina, invites applications and nominations for tenure-track faculty positions at the assistant professor level, to begin August 2012. We are interested in strong candidates in all active research areas of computer science, as well as interdisciplinary areas.

The department is committed to increasing the diversity of its faculty, and we strongly encourage applications from women and minority candidates.

A successful candidate must have a solid disciplinary foundation and demonstrate promise of outstanding scholarship in every respect, including research and teaching. Please refer to www.cs.duke.edu for information about the department and to www.provost.duke.edu/faculty/ for information about the advantages that Duke offers to faculty.

Applications should be submitted online through the link provided at www.cs.duke.edu/facsearch. A Ph.D. in computer science or related area is required. To guarantee full consideration, applications and letters of reference should be received by December 1, 2011.

Durham, Chapel Hill, and the Research Triangle of North Carolina are vibrant, diverse, and thriving communities, frequently ranked among the best places in the country to live and work. Duke and the many other universities in the area offer a wealth of education and employment opportunities for spouses and families.

Duke University is an affirmative action, equal opportunity employer.

Washington University in St. Louis
SCHOOL OF ENGINEERING & APPLIED SCIENCE

The DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING at WASHINGTON UNIVERSITY IN ST. LOUIS invites applications for the tenured position of Professor and Department Chair. The successful candidate will have a Ph.D. or equivalent in a related field, demonstrated continuing excellence in research, and a strong commitment to undergraduate teaching and postgraduate education. The chair will oversee significant growth of a department currently composed of 20 faculty and 80 doctoral students with numerous research interests including cyber-physical systems, computational biology and biomedical computing, human-centered and intelligent computing systems, networking and communications, and parallel computation (*http://www.cse.wustl.edu/*). The chair will conduct research, publish in peer-reviewed journals and conferences, teach relevant courses, advise students, and participate in University service.

Washington University (*http://www.wustl.edu/*) is a private university with roughly 6,000 full-time undergraduates and 6,000 graduate students. It is nationally known for the high quality of its student body and attractive campus, which borders residential neighborhoods and one of the nation's largest urban parks (*http://www.forestparkforever.org/*). Faculty enjoy the advantage of a Midwest cost of living, and many walk or bike to work.

Applicants should submit their curriculum vitae, a short summary of past research accomplishments and future plans, a teaching statement, and the names of at least three references electronically as a single PDF file to *csechairsearch@seas.wustl*.edu. Washington University is an AA/EOE and is strongly committed to enhancing the diversity of its faculty; minority and women scientists are especially encouraged to apply. Employment eligibility verification will be required upon employment.

Florida State University
Assistant Professor
Tenure-Track Assistant Professor Positions

The Department of Computer Science at the Florida State University invites applications for multiple tenure-track Assistant Professor positions to begin August 15, 2012. Positions are 9-mo, full-time, tenure-track, and benefits eligible. We encourage strong applicants in all areas of Computer Science to apply. Preference may be given to applicants with research experience in the areas of Databases and Security. Applicants should hold a PhD in Computer Science or closely related field, and have excellent research and teaching accomplishments/potential. The department offers degrees at the BS, MS, and PhD levels. The department is an NSA Center of Academic Excellence in Information Assurance Education (CAE/IAE) and Research (CAE-R).

FSU is classified as a Carnegie Research I university. Its primary role is to serve as a center for advanced graduate and professional studies while emphasizing research and providing excellence in undergraduate education. The department has experienced rapid growth in the major and new degree programs. Further information can be found at http://www.cs.fsu.edu

Screening will begin January 1, 2012 and will continue until the positions are filled. Please apply online with curriculum vitae, statements of teaching and research philosophy, and the names of five references, at http://www.cs.fsu.edu/positions/apply.html

Questions can be e-mailed to Prof. Mike Burmester, Chair Search Committee, recruitment@cs.fsu.edu or to Prof. Robert van Engelen, Department Chair, chair@cs.fsu.edu.

The Florida State University is a Public Records Agency and an Equal Opportunity/Access/Affirmative Action employer, committed to diversity in hiring.

Hawai'i Pacific University
Assistant/Associate/ Full Professor of Computer Science

Hawai'i Pacific University's Department of Mathematics and Computer Science invites applications for a career-track Assistant or Associate Professor, or Professor of Computer Science position to start in Spring or Fall, 2012.

Applicants should have a Ph.D., Ed.D., or D.B.A. in Computer Science, Information Systems, or a related field; and teaching experience at the university level. Preferred qualifications include experience as a program chair; experience with class scheduling, undergraduate and graduate curriculum development, assessment, funding, and recruiting; industry experience; and security expertise.

HPU is the largest private university in Hawai'i, with about 9000 students from over 100 countries. Information about the HPU computer science program can be found at www.hpu.edu/cs <http://www.hpu.edu/cs>.

Applicants should apply online at www.hpu.edu/employment. Please submit a letter of application, copies of transcripts, curriculum vitae, statements of teaching and research philosophy, list of publications, and three letters of recommendation that include an assessment of teaching abilities by either mail: Human Resources, Faculty Position in Computer Science, 1132 Bishop Street, Suite 310, Honolulu, HI 96813; by Fax (808) 544-1192; or by email: hr@hpu.edu.

We are proud to be an EEO/AA employer M/F/D/V. We maintain a drug-free workplace and perform pre-employment substance abuse testing.

In Accordance with the Jenne Clery Disclosure of Campus Security Policy and Campus Crime Statistics Act, annual campus crime statistics for Hawai'i Pacific University (HPU) may be viewed at: http://www.ope.ed.gov/security or a paper copy may be obtained upon request from HPU Campus Security or Administrative Services Office.

The Hong Kong University of Science and Technology
Department of Computer Science and Engineering
Faculty Positions

The Department of Computer Science and Engineering, HKUST (http://www.cse.ust.hk/) has more than 40 faculty members, recruited from major universities and research institutions around the world, and about 800 students (including about 200 postgraduate students). The medium of instruction is English. In 2011, we were ranked 21st among all Computer Science Departments worldwide according to Academic Ranking of World Universities, and 26th according to QS World University Ranking.

The Department will have at least two tenure-track faculty openings at Assistant Professor/Associate Professor/Professor levels for the 2012-2013 academic year. We are looking for faculty candidates with interests in bioinformatics, security, or cloud computing. Strong candidates in core computer science and engineering research areas will also be considered.

Salary is highly competitive and will be commensurate with qualifications and experience. Fringe benefits include medical/dental benefits and annual leave. Housing will also be provided where applicable. For appointment at Assistant Professor/Associate Professor level, initial appointment will normally be on a three-year contract. A gratuity will be payable upon completion of contract.

Applications should be sent through e-mail including a cover letter, curriculum vitae (including the names and contact information of at least three references), a research statement and a teaching statement (all in PDF format) to csrecruit@cse.ust.hk. Priority will be given to applications received by **29 February 2012**. Applicants will be promptly acknowledged through e-mail upon receiving the electronic application material.

(Information provided by applicants will be used for recruitment and other employment-related purposes.)

Louisiana Tech University
Research Assistant Professor, Non-Tenure Track

Initial appointment for one year; possibly extendable. PhD required in computer science, statis-

tics, mathematics, or related field. Opportunities for pursuing cutting edge research and joining a group of leading researchers. Background required in cyber security, internet, social networks, data mining, or related areas. Highly competitive salary. Contact Person: Dr. Vir V. Phoha, Email Address: phoha@latech.edu

Max Planck Institute for Informatics
Junior Research Group Leader

The Max Planck Institute for Informatics, as the coordinator of the Max Planck Center for Visual Computing and Communication (MPC-VCC), invites applications for

Junior Research Groups Leaders in the Max Planck Center for Visual Computing and Communication

The Max Planck Center for Visual Computing and Communications offers young scientists in information technology the opportunity to develop their own research program addressing important problems in areas such as image communication, computer graphics, geometric computing, imaging systems, computer vision, human machine interface, distributed multimedia architectures, multimedia networking, visual media security.

The center includes an outstanding group of faculty members at Stanford's Computer Science and El¬ectrical Engineering Departments, the Max Planck Institute for Informatics, and Saarland University.

The program begins with a preparatory 1-2 year postdoc phase (**Phase P**) at the Max Planck Institute for Informatics, followed by a two-year appointment at Stanford University (**Phase I**) as a visiting assistant professor, and then a position at the Max Planck Institute for Informatics as a junior research group leader (**Phase II**). However, the program can be entered flexibly at each phase, commensurate with the experience of the applicant.

Applicants to the program must have completed an outstanding PhD. Exact duration of the preparatory postdoc phase is flexible, but we typically expect this to be about 1-2 years. Applicants who completed their PhD in Germany may enter Phase I of the program directly. Applicants for Phase II are expected to have completed a postdoc stay abroad and must have demonstrated their outstanding research potential and ability to successfully lead a research group.

Reviewing of applications will commence on **November 1, 2011**. The final deadline is **December 31, 2011**. Applicants should submit their CV, copies of their school and university reports, list of publications, reprints of five selected publications, names of references, a brief description of their previous research and a detailed description of the proposed research project (including possible opportunities for collaboration with existing research groups at Saarbrücken and Stanford) to:

Prof. Dr. Hans-Peter Seidel
Max Planck Institute for Informatics,
Campus E 1 4, 66123 Saarbrücken, Germany;
Email: mpc-vcc@mpi-inf.mpg.de

The Max Planck Center is an equal opportunity employer and women are encouraged to apply.

Additional information is available on the website http://www.mpc-vcc.de

Max Planck Institute for Software Systems (MPI-SWS)
Tenure-track openings

Applications are invited for tenure-track and tenured faculty positions in all areas related to the study, design, and engineering of software systems. These areas include, but are not limited to, data and information management, programming systems, software verification, parallel, distributed and networked systems, and embedded systems, as well as cross-cutting areas like security, machine learning, usability, and social aspects of software systems. A doctoral degree in computer science or related areas and an

outstanding research record are required. Successful candidates are expected to build a team and pursue a highly visible research agenda, both independently and in collaboration with other groups. Senior candidates must have demonstrated leadership abilities and recognized international stature.

MPI-SWS, founded in 2005, is part of a network of eighty Max Planck Institutes, Germany's premier basic research facilities. MPIs have an established record of world-class, foundational research in the fields of medicine, biology, chemistry, physics, technology and humanities. Since 1948, MPI researchers have won 17 Nobel prizes. MPI-SWS aspires to meet the highest standards of excellence and international recognition with its research in software systems.

To this end, the institute offers a unique environment that combines the best aspects of a university department and a research laboratory:

a) Faculty receive generous base funding to build and lead a team of graduate students and post-docs. They have full academic freedom and publish their research results freely.

b) Faculty supervise doctoral theses, and have the opportunity to teach graduate and undergraduate courses.

c) Faculty are provided with outstanding technical and administrative support facilities as well as internationally competitive compensation packages.

MPI-SWS currently has 11 tenured and tenure-track faculty, and is funded to support 17 faculty and about 100 doctoral and post-doctoral positions. Additional growth through outside funding is possible. We maintain an open, international and diverse work environment and seek applications from outstanding researchers regardless of national origin or citizenship. The working language is English; knowledge of the German language is not required for a successful career at the institute.

The institute is located in Kaiserslautern and Saarbruecken, in the tri-border area of Germany, France and Luxembourg. The area offers a high standard of living, beautiful surroundings and easy access to major metropolitan areas in the center of Europe, as well as a stimulating, competitive and collaborative work environment. In immediate proximity are the MPI for Informatics, Saarland University, the Technical University of Kaiserslautern, the German Center for Artificial Intelligence (DFKI), and the Fraunhofer Institutes for Experimental Software Engineering and for Industrial Mathematics.

Qualified candidates should apply online at http://www.mpi-sws.org/application. The review of applications will begin on January 3, 2012, and applicants are strongly encouraged to apply by that date; however, applications will continue to be accepted through January 2012.

The institute is committed to increasing the representation of minorities, women and individuals with physical disabilities in Computer Science. We particularly encourage such individuals to apply.

Michigan Technological University
Department of Computer Science
Tenure-Track Faculty Position

Applications are invited for a tenure-track faculty position at the assistant professor level beginning August 2012. Candidates are expected to have a Ph.D. in computer science or a closely related field and to demonstrate potential for excellence in teaching and research. Candidates with research interests in human-centered computing, including human-computer interaction, software engineering, information integration and informatics, and health-related computing, will receive the strongest consideration; however, excellent candidates in other areas will be considered.

The Department has 320 undergraduates in three degree programs (Computer Science, Computer Systems Science and Software Engineering) and approximately 50 M.S. and Ph.D. students. One of the Department's current goals is to increase diversity among its undergraduate majors. Faculty research interests include artificial intelligence, bioinformatics, computer architecture, compilers, distributed systems, embedded and multimedia systems, graphics and visualization, human computer interaction, parallel computing, security and software engineering. In addition, the Department has a central role in the interdisciplinary Computational Science and Engineering Ph.D. program which fosters research and teaching in the application of computer science to engineering and scientific problems.

The Department is housed in Rekhi Hall which includes space for graduate student and faculty offices, instructional labs, and classrooms. The facility also houses faculty research labs, including space for human-computer interaction, virtual reality, artificial intelligence, graphics and visualization, compilers and architecture, and computer security.

Michigan Technological University is a research university with approximately 7,000 students and 400 faculty with educational and research programs in computing, engineering, physical and social sciences, forestry, humanities and business. Michigan Tech is located in Michigan's scenic Upper Peninsula and is surrounded by Lake Superior and nearby forests. The community offers year-round recreational and cultural opportunities. This environment, combined with a competitive compensation package and a low cost of living, results in an excellent quality of life.

Review of applications will continue until the position is filled. For full consideration, applications should be received by December 2, 2011. Women and minorities are particularly encouraged to apply. Applicants should send a resume, email address, a teaching statement, a research statement and a list of at least three references to:

Steven M. Carr, Chair
Department of Computer Science
Michigan Technological University
Houghton, Michigan 49931
cssearch@mtu.edu
(906) 487-2209

For more information see our web page
http://www.mtu.edu/cs/.

In addition to the present search, a search to fill ten growth positions in "Transportation" and "Water" are under way and qualified candidates are encouraged to send a separate application, following the "How to Apply" guidelines at www.mtu.edu/sfhi.

Michigan Tech is also an ADVANCE institution, one of a select number of universities in receipt of NSF funds in support of our commitment to increase diversity and the participation and advancement of women in STEM. For more information, please visit our ADVANCE web page at http://advance.mtu.edu/.

Michigan Technological University is an equal opportunity educational institution/equal opportunity employer.

Mississippi State University

Faculty Position in Computer Science and Engineering

The Department of Computer Science and Engineering (http://www.cse.msstate.edu) is seeking to fill an open position for a tenure-track faculty member at the Assistant/Associate Professor levels. Evidence of strong potential for excellence in research (including attracting external funding) and teaching at the graduate and undergraduate levels is required. The primary research areas of interest are artificial intelligence, bioinformatics, and computer security. Other areas will also be considered.

Mississippi State University has approximately 1300 faculty and 20,000 students. The Department of Computer Science and Engineering has 17 tenure-track faculty positions and offers academic programs leading to the bachelor's, master's and doctoral degrees in computer science and bachelor's degrees in software engineering and computer engineering. Faculty members and graduate students work with a number of on-campus research centers including the Critical Infrastructure Protection Center, the High Performance Computing Collaboratory, the Institute for Imaging and Analytical Technologies, the Institute for Genomics, Biocomputing, and Biotechnolgy, the Center for Advanced Vehicular Systems, and the Geosystems Research Institute. Department research expenditures total around 5.2 million dollars per year.

Candidates for this position are expected to hold a PhD in computer science or closely related field (ABDs may be considered). Level of appointment is commensurate with qualifications and experience.

Applicants should submit a letter of application, curriculum vita, teaching statement, research statement, and names and contact information of at least three references online at http://www.jobs.msstate.edu/. Review of applications will begin not earlier than December 2011 and continue until the position is filled. MSU is an Affirmative Action/Equal Opportunity Employer.

Princeton University

Computer Science Department
Tenure-Track Positions
Assistant Professor

The Department of Computer Science at Princeton University invites applications for faculty positions at the Assistant Professor level. We are accepting applications in all areas of Computer Science.

Applicants must demonstrate superior research and scholarship potential as well as teaching ability. A PhD in Computer Science or a related area is required. Successful candidates are expected to pursue an active research program and to contribute significantly to the teaching programs of the department. Applicants should include a CV and contact information for at least three people who can comment on the applicant's professional qualifications.

There is no deadline, but review of applications will start in December 2011. Princeton University is an equal opportunity employer and complies with applicable EEO and affirmative action regulations. You may apply online at: http://jobs.cs.princeton.edu/.

Requisition Number: 0110422

Rutgers University

Department of Computer Science
Tenure-Track Position

The Department of Computer Science at Rutgers University invites applications for tenure-track faculty positions at the rank of Assistant, Associate or full Professor, with appointments starting in September 2012, subject to the availability of funds. All areas in experimental computer systems will be considered, but special emphasis will be given to compilers and programming languages.

Applicants for this research/teaching position must, at minimum, be in the process of completing a dissertation in Computer Science or a closely related field, and should show evidence of exceptional research promise, potential for developing an externally funded research program, and commitment to quality advising and teaching at the graduate and undergraduate levels. Hired candidates who have not defended their Ph.D. by September will be hired at the rank of Instructor, and must complete the Ph.D. by December 31, 2012 to be eligible for tenure-track title retroactive to start date.

Applicants should go to http://www.cs.rutgers.edu/employment/ and submit their curriculum vitae, a research statement addressing both past work and future plans and a teaching statement along with three letters of recommendation. If electronic submission is not possible, hard copies of the application materials may be sent to:

Mary Hoffman
Computer Science Department
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854

Applications should be received by December 20, 2011 for full consideration.

Rutgers subscribes to the value of academic diversity and encourages applications from individuals with varied experiences, perspectives, and backgrounds. Females, minorities, dual-career couples, and persons with disabilities are encouraged to apply.

Rutgers is an affirmative action/equal opportunity employer.

Stanford University

Department of Computer Science
Assistant or untenured Associate Professor

The Department of Computer Science at Stanford University invites applications for tenure-track faculty positions at the junior level (Assistant or untenured Associate Professor). We give higher priority to the overall originality and promise of the candidate's work than to the candidate's sub-area of specialization within Computer Science.

We are seeking applicants from all areas of Computer Science, spanning theoretical foundations, systems, software, and applications. We are also interested in applicants doing research at the frontiers of Computer Science with other disciplines, especially those with potential connections to Stanford's main multidisciplinary initiatives: Energy, Human Health, Environment and Sustainability, the Arts and Creativity, and the International Initiative. Interdisciplinary candidates whose research combines other fields of engineering or mathematics with computer science may be considered for a joint appointment in the Institute for Computational and Mathematical Engineering (http://icme.stanford.edu/).

Applicants must have completed (or be completing) a Ph.D., must have demonstrated the ability to pursue a program of research, and must have a strong commitment to graduate and undergraduate teaching. A successful candidate will be expected to teach courses at the graduate and undergraduate levels, and to build and lead a team of graduate students in Ph.D. research. Further information about the Computer Science Department can be found at http://cs.stanford.edu. The School of Engineering website may be found at http://soe.stanford.edu.

Applications should include a curriculum vita, brief statements of research and teaching interests, and the names and contact information of at least four references. Applications should be sent to: http://soe-apps.stanford.edu/FacultyApplyCS

Questions should be directed to, Search Committee Chair, c/o Laura Kenny-Carlson, via electronic mail to search@cs.stanford.edu.

The review of applications will begin on November 28, 2011, and applicants are strongly encouraged to submit applications by that date; however, applications will continue to be accepted at least until February 15, 2012.

Stanford University is an equal opportunity employer and is committed to increasing the diversity of its faculty. It welcomes nominations of and applications from women and members of minority groups, as well as others who would bring additional dimensions to the university's research and teaching missions.

Stanford University

Graduate School of Business
Tenure-Track Positions

Faculty Positions in Operations, Information and Technology The Operations, Information and Technology (OIT) area at the Graduate School of Business, Stanford University, is seeking qualified applicants for full-time, tenure-track posi-

tions, starting in the 2012-2013 academic year. All ranks and relevant disciplines will be considered. Applicants are considered in all areas of Operations, Information and Technology (OIT) that are broadly defined to include the analytical and empirical study of technological systems, in which technology, people, and markets interact. It thus includes operations, information systems/technology, and management of technology. Applicants are expected to have rigorous training in management science, engineering, computer science, economics, and/or statistical modeling methodologies. The appointed will be expected to do innovative research in the OIT field, to participate in the school's PhD program and to teach both required and elective courses in the MBA program. Junior applicants should have or expect to complete a PhD by September 1, 2012.

Applicants are ***strongly encouraged*** to submit their applications electronically by visiting the web site http://www.gsb.stanford.edu/recruiting and uploading their curriculum vitae, research papers and publications, and teaching evaluations, if applicable, on that site. Alternatively, all materials may be sent by e-mail to faculty_recruiter@gsb.stanford.edu, or by postal mail (non-returnable) to Office of Faculty Recruiting, Graduate School of Business, Stanford University, 655 Knight Way Way, Stanford, CA 94305-7298. However, submissions via e-mail and postal mail can take 4-6 weeks for processing. For an application to be considered complete, each applicant must have three letters of recommendation emailed to the preceding email address, or sent via postal mail. **The application deadline is November 15, 2011.**

Stanford University is an equal opportunity employer and is committed to increasing the diversity of its faculty. It welcomes nominations of and applications from women and members of minority groups, as well as others who would bring additional dimensions of diversity to the University's research, teaching and clinical missions.

Texas State University-San Marcos
Department of Computer Science

Applications are invited for a faculty position at the rank of Assistant, Associate, or full Professor to start on September 1, 2012. Consult the department's recruiting page, http://www.cs.txstate.edu/recruitment/, for job duties, qualifications, application procedures, and information about the university and the department.

Texas State University-San Marcos will not discriminate against any person in employment or exclude any person from participating in or receiving the benefits of any of its activities or programs on any basis prohibited by law, including race, color, age, national origin, religion, sex, disability, veterans' status, or on the basis of sexual orientation. Texas State University-San Marcos is a member of the Texas State University System.

University at Buffalo, The State University of New York
Faculty Positions in Computer Science and Engineering

Tak Associate/Assistant Professor position: The CSE Department invites excellent candidates in all core areas of Computer science and Engineering to apply for an opening at the associate/assistant professor level. This is an endowed position with five years research funding of $100K per year.

Assistant Professor positions: The CSE Department also invites excellent candidates in all core areas of Computer science and Engineering to apply for openings at the assistant professor level.

The department is affiliated with successful centers devoted to biometrics, bioinformatics, biomedical computing, cognitive science, document analysis and recognition, high performance computing, and information assurance.

Candidates are expected to have a Ph.D. in Computer Science/Engineering or related field by August 2012, with an excellent publication record and potential for developing a strong funded research program.

Applications should be submitted by December 31, 2011 electronically via http://www.ubjobs.buffalo.edu/.

The University at Buffalo is an
Equal Opportunity Employer/Recruiter.

University of California, Santa Barbara
Faculty Position in Computer Science
Tenure-Track position

The Department of Computer Science at the University of California, Santa Barbara, invites applications for a tenure-track position effective August 2012. We are particularly interested in outstanding candidates in the areas of applied cryptography and system security; however, exceptional candidates in all areas of computer science will be considered.

The Department of Computer Science has grown rapidly, both in size and stature, over the past 10 years, accompanied by a five-fold increase in extramural funding. The department, with 30 faculty and more than 100 doctoral students, is part of the College of Engineering, which is ranked among the top 20 in the Nation by the 2008 US News and World Report. The PhD program of the Department of Computer Science has recently been ranked among the top 10 departments in the nation by the National Research Council (NRC).

Additional information about the department and our graduate program can be found at http://www.cs.ucsb.edu. Applicants are expected to hold a doctoral degree in Computer Science or related fields, show outstanding research potential, and have a strong commitment to teaching.

Primary consideration will be given to applications received by December 15, 2011; however, the position will remain open until filled. Applications should be submitted electronically as PDF documents to: https://www.cs.ucsb.edu/recruit/faculty. Applications must include a detailed resume, research and teaching statements, and the names and addresses of four references.

The Department is especially interested in candidates who can contribute to the diversity and excellence of the academic community through research, teaching, and service. We are an Equal Opportunity/Affirmative Action employer.

University of Central Missouri
Assistant Professor of Computer Science-Tenure Track

UCM invites applications for a tenure track position in Computer Science beginning August, 2012. Ph.D. in CS or a closely related field is required. Preference given to candidates with expertise in computer security or software engineering. Screening of applications begins Nov. 15, 2011, and continues until position is filled. For more information, go to http://www.ucmo.edu/math-cs/openings.cfm

University of Illinois at Chicago
Faculty Position
Department of Computer Science, UIC

The Computer Science Department at the University of Illinois at Chicago invites applications for tenure-track positions at the rank of Assistant Professor (exceptional candidates at other ranks may also be considered). Candidates in the following areas are especially encouraged to apply: Computer Security, Software Engineering, Machine Learning and Computer Systems (Mobile/Ubiquitous computing and other experimental subareas).

The University of Illinois at Chicago (UIC) ranks among the nation's top 50 universities in federal research funding. It is the largest research university in the Chicago area, and is one of the most diverse universities in the country. The Computer Science department has 27 tenure-track faculty representing major areas of computer science, and offers BS, MS and PhD degrees. Two of our faculty members are ACM Fellows and eight members are recipients of NSF CAREER awards. Our annual research funding has averaged $6.5M over the last five years and includes major funding from NSF, DARPA, DoD and NASA, including two NSF IGERT awards, nine Trustworthy Computing awards and several other research and instrumentation grants; awards from state agencies such as the Illinois Department of Transportation, and from companies such as Google, Yahoo! and Motorola. Our department is home to many pioneering and discipline-defining efforts in the areas of virtual reality (CAVE), software engineering (Petri Nets, Model Checking), Data Management and Mining, and Computational Transportation. We have growing research programs in areas such as computational biology, learning technologies, mobile and distributed systems, and security and privacy. At UIC, there are plenty of opportunities for interdisciplinary work—UIC houses the largest medical school in the country, and our faculty are engaged with several cross-departmental collaborations with faculty from health sciences, social sciences and humanities, urban planning and the business school.

Chicago is the third most populous city in the USA. Located by the shore of Lake Michigan, the city offers an outstanding array of cultural and culinary experiences. As the birthplace of the modern skyscraper, Chicago boasts one of the world's tallest and densest skylines, combined with an extensive system of parks and public transit. Its primary airport is the second busiest in the world, with frequent non-stop flights to virtually anywhere. Yet the cost of living, whether in an 85th floor condominium downtown or on a tree-lined street in one of the nation's finest school dis-

tricts, is surprisingly low.

Applications must be submitted at https://jobs.uic.edu/. Please include a resume, teaching and research statements, and names and addresses of at least three references in the online application. Applicants needing additional information may contact the Faculty Search Chair at search@cs.uic.edu.

Application processing will commence on Nov 15th. We will continue to accept and process applications after that date until all the positions are filled. The University of Illinois at Chicago is an Affirmative Action/Equal Opportunity Employer.

University of Illinois at Springfield

Assistant Professor

The Computer Science Department at the University of Illinois Springfield (UIS) invites applications for a beginning assistant professor, tenure track position to begin January, 2012. A Ph.D. in Computer Science or closely related field is required. The position involves graduate and undergraduate teaching, supervising student research, and continuing your research. Many of our classes are taught online. All areas of expertise will be considered, but computer security is of special interest to the Department. Review of applications will begin on November 21, 2011 and continue until the position is filled or the search is terminated. Please send your vita and contact information for three references to Chair Computer Science Search Committee; One University Plaza; UHB 3100; Springfield, IL 62703-5407.

Located in the state capital, the University of Illinois Springfield is one of three campuses of the University of Illinois. The UIS campus serves approximately 5,000 students in 23 undergraduate and 21 graduate degree programs. The academic curriculum of the campus emphasizes a strong liberal arts core, an array of professional programs, extensive opportunities in experiential education, and a broad engagement in public affairs issues of the day. The campus offers many small classes, substantial student-faculty interaction, and a rapidly evolving technology enhanced learning environment. Its diverse student body includes traditional, non-traditional, and international students. Twenty-five percent of majors are in 17 undergraduate and graduate online degree programs and the campus has received several national awards for its implementation of online learning. UIS faculty are committed teachers, active scholars, and professionals in service to society. You are encouraged to visit the university web page at http://www.uis.edu and the department web page at http://csc.uis.edu . UIS is an affirmative action/equal opportunity employer with a strong institutional commitment to recruitment and retention of a diverse and inclusive campus community. Women, minorities, veterans, and persons with disabilities are encouraged to apply.

University of Maryland, Baltimore County (UMBC)

CSEE Department

Assistant Professor

The Univ. of Maryland, Baltimore County (UMBC) Computer Science and Electrical Engineering Dept. invites applications for one or more tenure track positions to begin Aug. 2012. All Computer Science areas will be considered. We are especially interested in candidates with systems/experimental approaches. For more information, see http://cs.umbc.edu/about/jobs/.

UMBC is an AA/EOE.

University of Maryland, College Park

Center for Bioinformatics and Computational Biology

Professor and Director

The University of Maryland invites applications for Director of the Center for Bioinformatics and Computational Biology. Candidates are expected to be prominent scholars with publications and research experience at the interface of biological science and computing. Their primary responsibility will be to lead a nationally visible research program complementing existing strengths in computational genomics, proteomics, and molecular evolution. They will also be expected to promote the CBCB, and help build collaborative relationships, both on and off-campus. Information about the Center can be found at www.cbcb.umd.edu. Collectively, the CBCB faculty spans the fields of computer science, mathematics and statistics, biology, and biochemistry. The Center is housed in contiguous space and has access to significant high-end computing infrastructure through the University of Maryland Institute for Advanced Computer Studies. CBCB faculty members are also affiliated with at least one other campus academic unit appropriate to their interests. There is ample potential for collaboration with other organizations in the area, such as the NIH, the JCVI, and the Smithsonian Institution. For more information contact the search chair, Thomas D. Kocher (tdk@umd.edu). To apply, send a letter of application, curriculum vitae, and names of three references, following the instructions at http://cbcb.umd.edu/hiring/. Review of applications will begin November 15, 2011.

The University of Maryland is an affirmative action, equal opportunity employer. Women and minorities are encouraged to apply.

University of Massachusetts Lowell

Computer Science Department

Tenure-Track Assistant/Associate/ Full Professor

The University of Massachusetts Lowell is 25 miles northwest of Boston in the high-tech corridor of Massachusetts. UMass Lowell is classified as a DRU/H University. The Computer Science Department has 14 tenured and tenure-track faculty and one full-time lecturer, serving 250 BS students, 110 MS students, and 55 PhD students. We are looking to fill a number of tenure-track/tenured positions in the next few years. The Department has received approximately $6M in external research funding in the past two years and has four NSF Career Award recipients. More information about the Department can be found at www.cs.uml.edu.

We invite applications for two tenure-track positions at any rank to start in January 2012 or September 2012. Appointments with tenure at the rank of Associate Professor or higher could be made based on qualifications. The successful candidate will be expected to teach undergraduate and graduate courses, including Department core and specialty areas based on the candidate's expertise, conduct external funded research, direct PhD dissertations, and provide service to the Department and the University.

Qualifications: applicants must have a PhD in computer science or a closely related discipline and must be committed to developing and sustaining externally funded research programs. By the time of appointment, applicants must either have one or more years of teaching and research experience as an assistant professor in a US University or have one or more years as a post-doctoral researcher in a US University or research lab. All applicants should have participated in significant federal grant writing. Current US federal funding as PI is highly desirable. Outstanding candidates in any major computer science research area will be considered.

For appointment at the rank of Associate Professor or higher, applicants must have substantial research, teaching, and service experience, have made significant contributions to their fields with strong ongoing research projects, and be the PI of substantial current funding from major US funding agencies.

Interested applicants should apply online at https://jobs.uml.edu.

The University of Massachusetts Lowell is committed to increasing diversity in its faculty, staff, and student populations, as well as curriculum and support programs, while promoting an inclusive environment. We seek candidates who can contribute to that goal and encourage you to apply and to identify your strengths in this area.

The University of Michigan, Ann Arbor

Department of Electrical Engineering and Computer Science

Computer Science and Engineering Division Faculty Positions

Applications and nominations are solicited for multiple faculty positions in the Computer Science and Engineering (CSE) Division.

1. As part of an interdisciplinary cluster hire, CSE seeks individuals broadly interested in parallel systems that scale to petascale and beyond. Relevant areas include run-time systems, compilers, algorithms, programming languages, tools, applications, and networking.

2. As part of an interdisciplinary cluster hire, CSE seeks individuals in the area of Computational Media and Interactive Systems. Relevant research areas include those related to analyzing, understanding, representing, and creating computational media, including but not limited to audio, music, image, and video information retrieval, video understanding and mashups, time-based, distributed, collaborative and interactive media, and automatic classification, clustering and activity recognition in heterogeneous and social media.

3. CSE also seeks individuals in the broad areas of theoretical computer science and software systems (for the latter including but not limited to the areas of programming languages, databases and networking).

We encourage all highly qualified candidates to apply, particularly those with expertise in the above areas. Qualifications include an outstanding academic record, a doctorate or equivalent in computer engineering or computer science or a discipline relevant to the positions, and a strong commitment to teaching and research. Applications must be received by January 1, 2012.

To apply, please complete the form at: http://www.eecs.umich.edu/eecs/jobs/csejobs.html

Electronic applications are strongly preferred, but you may alternatively send resume, teaching statement, research statement and names of three references to:

Professor Satinder Singh Baveja,
Chair, CSE Faculty Search
Department of Electrical Engineering
and Computer Science
University of Michigan
2260 Hayward Street,
Ann Arbor, MI 48109-2121

The University of Michigan is a Non-Discriminatory/ Affirmative Action Employer with an Active Dual-Career Assistance Program. The college is especially interested in candidates who can contribute, through their research, teaching, and/or service, to the diversity and excellence of the academic community.

University of Pennsylvania

Faculty Positions in Market and Social Systems

The University of Pennsylvania seeks outstanding individuals for a tenure-track or tenured faculty position in the Department of Electrical and Systems Engineering to start July 1, 2012. Applicants must have a Ph.D. in Engineering, Applied Mathematics, Computer Science, Operations Research, or equivalent. This search focuses on candidates working at the mathematical and computational foundations of Market and Social Systems Engineering - the formalization, analysis, optimization, and realization of interconnected systems that increasingly integrate engineering, computational, social and economic systems and methods. We are particularly interested in candidates who have a vision and interest in defining the research frontier and education of next-generation leaders and will contribute to the formation of a new interdisciplinary undergraduate program in this interdisciplinary field. The University of Pennsylvania is pioneering a new undergraduate program on Market and Social Systems Engineering (MKSE) http://www.mkse.upenn.edu, as an interdisciplinary effort focused on topics at the burgeoning intersection of systems engineering, operations research, game theory and mechanism design, algorithmic aspects of economics and sociology, and many related areas. The program is designed to embrace the emerging scientific and engineering principles underlying phenomena as diverse as network science, social networking, web search and its monetization, electronic commerce, modern financial networks, and many others.

The University seeks individuals with exceptional promise for, or proven record of, research achievement who will excel in teaching undergraduate and graduate courses and take a position of international leadership in defining their field of study. Leadership in cross-disciplinary collaborations is of particular interest.

Interested persons should submit an application by completing the form located on the Faculty Recruitment Web site at http://www.ese.upenn.edu/opps/ including curriculum vitae, and the names of at least three references.

The University of Pennsylvania is an Equal Opportunity Employer. Minorities/Women/ Individuals with Disabilities/Veterans are encouraged to apply.

University of South Florida

Assistant Professor

The Department of Mathematics and Statistics at the University of South Florida invites applications in Pure and Applied Discrete Mathematics for a tenure-track Assistant Professor position. For details see http://www.math.usf.edu/about/employ/.

Valdosta State University

Assistant Professor of Computer Science

Applications are invited for a ten-month tenure-track faculty position at the rank of Assistant Professor with starting date of August 1, 2012. Responsibilities include teaching at the undergraduate level, scholarly activity, and service to both the department and the university.

Applicants must complete a doctorate in Computer Science, Computer Information Systems, or a closely related field by August 2012 and should have a commitment to undergraduate teaching and should be able to contribute to curriculum development.

Valdosta State University, a multipurpose regional university within the University System of Georgia, has an enrollment of approximately 13,000 students. The Department of Mathematics and Computer Science offers Bachelor of Science degrees in Computer Science (CS) and Computer Information Systems (CIS), and has over 200 CS/CIS majors. The department has 34 full-time faculty members.

Contact Person: Dr. Ashok Kumar
Email Address: akumar@valdosta.edu
Apply URL:
http://www.valdosta.edu/academic/documents/Faculty_App2010.pdf

Virginia Tech

Electrical & Computer Engineering
Postdoctoral Associate

Postdoctoral Associate positions are available in Virginia Tech's ECE Department on a project involving software transactional memory. Applicants with strong systems backgrounds (compilers, virtual machines, operating systems, etc.) are sought. Apply to http://jobs.vt.edu, posting 011079.

Washington State University Vancouver

Computer Science Faculty
Assistant Professor Level

COMPUTER SCIENCE FACULTY – Washington State University Vancouver invites applications for a tenure-track position at the assistant professor level beginning 8/16/2012. Candidates are sought with expertise in software engineering, **software architecture and design, or software security**.

Required qualifications: Ph.D. in Computer Science or Computer Engineering at the time of employment and demonstrated ability to (1) develop funded research program, (2) establish strong industrial collaborations, and (3) teach undergraduate/graduate courses. Preferred qualifications: knowledge of the ABET accreditation process and relevant industrial background.

WSU Vancouver serves about 3000 graduate and undergraduate students and is **fifteen miles north of Portland, Oregon**. The rapidly growing School of Engineering and Computer Science (ENCS) equally values both research and teaching. WSU is Washington's land grant university with faculty and programs on four campuses. For more information: http://www.vancouver.wsu.edu/encs. WSU Vancouver is committed to building a culturally diverse educational environment.

Applications must include: (1) cover letter with a clear description of experience relevant to the position; (2) vita including a list of references; and (3) **maximum three-page total** summary addressing the following: (a) How your research will expand or complement the current research activities in ENCS; (b) A list of the existing ENCS courses and proposed new courses that you can develop/teach; and (c) Your past experience or future plans on working with diverse student and community populations. Application deadline is **December 5, 2011**. Mail application materials to CS Search Committee, School of ENCS - VELS 130, Washington State University, 14204 NE Salmon Creek Avenue, Vancouver, WA 98686-9600. WSU is committed to excellence through diversity, has faculty friendly policies including a partner accommodation program, and a NSF ADVANCE Institutional Transformation grant (see http://www.excelinse.wsu.edu/). WSU employs only US citizens and lawfully authorized non-citizens. WSU is an EO/AA educator and employer.

York University

Department of Computer Science and Engineering
Canada Research Chair (Tier 2)
Faculty Appointment

York University, Toronto: The Department of Computer Science and Engineering invite applications for Canada Research Chair (Tier 2) faculty appointment in Digital Media with research expertise in Data/ Information/ Scientific Visualization and/or Interaction Design at the Assistant/Associate Professor level in the tenure track stream. The deadline for the applications is November 30, 2011 with a start date of July 1, 2012. For detailed information, please visit http://yorku.ca/acadjobs. York University is an Affirmative Action Employer.

DOI:10.1145/2018396.2018422

Peter Winkler

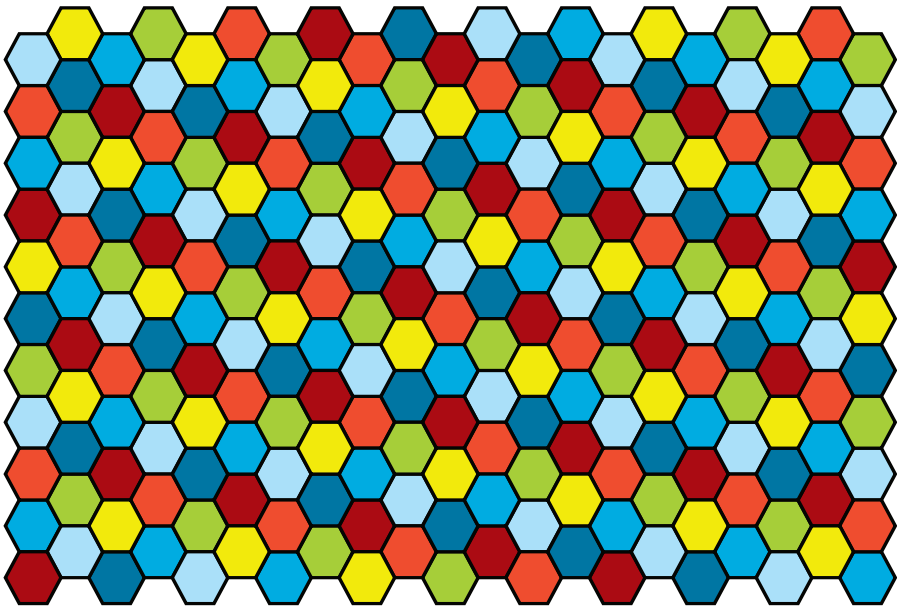
Puzzled
Distances Between Points on the Plane

Welcome to three new puzzles. Solutions to the first two will be published next month; the third is as yet (famously) unsolved.

The theme is distances between points on the plane. You might want a ruler and a large blank sheet of paper first...

1. It seems that in 1539, when Friar Marcos de Niza reported he had seen the fabled Seven Cities of Gold (in what is today New Mexico) he wasn't believed. According to my sources, he claimed the cities were located in such a way that among any three of them, at least two were exactly 10 leagues apart. Spanish officials claimed no such layout was possible on a flat surface. Were they right?

2. In more modern times, we would like to place nine equally strong Frisbee throwers in a field in such a way that no two of them are more than 100 yards apart, but as many pairs as possible are exactly that distance. How would you place them? Can you prove you can't do better?



Seven-coloring of the plane. If the hexagons' sides are of length a bit less than 1/2, no two points at distance 1 will have the same color.

3. You can use your solution to the first puzzle to show it will take at least four colors to paint the plane in such a way that no two points at unit distance get the same color. On the other hand, if you tile the plane with regular hexagons of the right size and paint them with seven colors in such a way that each hexagonal cell is surrounded by six cells of the other six colors, then you will have a way to paint the plane with seven colors such that no two points at unit distance get the same color (see the figure here).

So, four colors are necessary, and seven sufficient, to color the plane in such a way that no two points at distance 1 are the same color. What's the right number? (To learn much more about this problem, read *The Mathematical Coloring Book* by Alexander Soifer, Springer, 2009.)

Meanwhile, all readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science at Dartmouth College, Hanover, NH.