

Can computers think?

Why is time moving so quickly?

Is there life beyond death?

What is the meaning of life?

How much of compassion is just posturing?

Has your greatest fear ever come true?



IEEE 9th World Congress on Services (SERVICES 2013)

June 27—July 2, 2013, Santa Clara Marriott, CA, USA (center of Silicon Valley)
Federation of 5 Theme Topic Conferences on Services from Different Angles
(<http://www.servicescongress.org>)



IEEE 6th International Conference on Cloud Computing (CLOUD 2013)

Cloud Computing is becoming a scalable services delivery and consumption platform in the field of Services Computing. The technical foundations of Cloud Computing include Service-Oriented Architecture and Virtualizations. Major topics cover Infrastructure Cloud, Software Cloud, Application Cloud, Social Cloud, & Business Cloud. Visit <http://thecloudcomputing.org>.

IEEE 20th International Conference on Web Services (ICWS 2013)

ICWS 2013 will feature data-centric, web-based services modeling, design, development, publishing, discovery, composition, testing, QoS assurance, adaptation, and delivery technologies and standards. Visit <http://icws.org>.



IEEE 10th International Conference on Services Computing (SCC 2013)

SCC 2013 will focus on services innovation lifecycle e.g., enterprise modeling, business consulting, solution creation, services orchestration, optimization, management, business process integration and management. Visit <http://conferences.computer.org/>



IEEE 2nd International Conference on Mobile Services (MS 2013)

MS 2013 will feature all aspects of mobile services including modeling, construction, deployment, middleware, and user experience with a special emphasis on context-awareness in social settings. Visit <http://themobileservices.org/2013>.



IEEE 2nd International Conference on Services Economics (SE-BigData 2013)

SE 2013 will focus on quantitative analysis of impact on service financial and enterprise operational outcome from BigData initiative, including outcome metrics identification, risk assessment and trade-off of IT solution selection and optimization of IT expense. Visit <http://ieeese.org/2013>.



Sponsored by IEEE Technical Committee on Services Computing (TC-SVC, <http://tab.computer.org/tscs>)

Conference proceedings are EI indexed. Extended versions of invited ICWS/SCC/CLOUD/MS/SE papers will be published in IEEE Transactions on Services (TSC, SCI & EI indexed), International Journal of Web Services Research (JWSR, SCI & EI indexed), International Journal of Business Process Integration and Management (IJBPIIM), and IEEE IT Pro (SCI & EI Indexed).



IBM Research



Submission Deadlines

ICWS 2013: 1/28/2013
CLOUD 2013: 1/28/2013
SCC 2013: 2/11/2013
MS 2013: 2/11/2013
SE 2013: 3/1/2013
SERVICES 2013: 3/1/2013

Contact: Liang-Jie Zhang (LJ)
zhanglj@ieee.org
(Steering Committee Chair)



IEEE TRANSACTIONS ON
**SERVICES
COMPUTING**



ACM Inroads

The magazine for computing educators worldwide



<http://inroads.acm.org>

Paving the way toward excellence in computing education

Departments

- 5 **Letter from ACM's History Committee Chair**
Understanding ACM's Past
By Mary Hall
-
- 7 **From the President**
Computer Science Revisited
By Vinton G. Cerf
-
- 8 **Letters to the Editor**
Why Open Access?
-
- 12 **BLOG@CACM**
Levels of Abstraction; Pre-Teens and Career Choices
Mark Guzdial writes about the need for programming languages to support multimedia at all levels. Judy Robertson shares insights about 12-year-old students' lack of understanding about computer science.
-
- 29 **Calendar**
-
- 113 **Careers**

Last Byte

- 126 **Puzzled**
Solutions and Sources
By Peter Winkler
-
- 128 **Q&A**
As Good As It Gets
Sanjeev Arora talks about proof, intractability, and finding the best way to approximate problems.
By Leah Hoffmann

News

- 15 **Quantum Quests**
Three breakthrough experiments involving photons have extended coherence times and indicated scalable production.
By Gregory Goth
-
- 18 **Zoom In, Zoom Out**
Drilling down to more detail on a computer screen, or moving out to see the context, is basic. But it's hardly simple and, after 20 years, innovations are still occurring.
By Gary Anthes
-
- 20 **In the Year of Disruptive Education**
As college tuitions soar, various online models vie to educate college students worldwide—at no cost.
By Paul Hyman

Viewpoints

- 24 **Emerging Markets**
IT Innovation for the Bottom of the Pyramid
New ways to develop technologies for the emerging growth markets.
By Richard Heeks
-
- 28 **Historical Reflections**
Saving Private Gromit
Reflections on the legalities and economics of preserving animations and games in Europe.
By David Anderson
-
- 31 **Kode Vicious**
Can More Code Mean Fewer Bugs?
The bytes you save today may bite you tomorrow.
By George V. Neville-Niel
-
- 33 **The Profession of IT**
Moods
Recognizing and working with moods—your own, your team's, and your customers'—is essential to professional success.
By Peter J. Denning
-
- 36 **Broadening Participation**
Why We Need an ACM Special Interest Group for Broadening Participation
A proposal for an international group focused on broadening participation.
By Teresa A. Dahlberg
-
- 39 **Viewpoint**
Alan Turing Remembered
A unique firsthand account of formative experiences with Alan Turing.
By William Newman



Practice



- 42 **The Essence of Software Engineering: The SEMAT Kernel**
A thinking framework in the form of an actionable kernel.
By Ivar Jacobson, Pan-Wei Ng, Paul E. McMahon, Ian Spence, and Svante Lidman

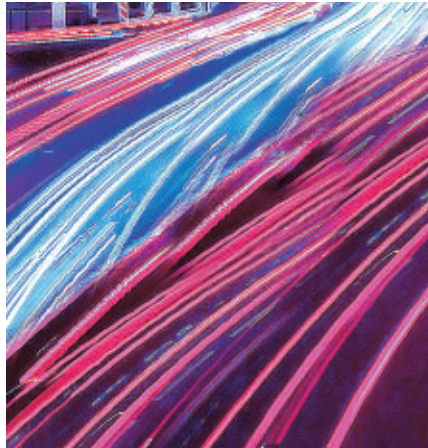
- 50 **Sender-side Buffers and the Case for Multimedia Adaptation**
A proposal to improve the performance and availability of streaming video and other time-sensitive media.
By Aiman Erbad and Charles "Buck" Krasic

- 59 **Anatomy of a Solid-State Drive**
While the ubiquitous SSD shares many features with the hard-disk drive, under the surface they are completely different.
By Michael Cornwell



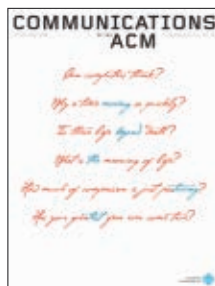
Articles' development led by acmqueue.queue.acm.org

Contributed Articles



- 64 **SPDYing Up the Web**
Improved performance and a proven deployment strategy make SPDY a potential successor to HTTP.
By Bryce Thomas, Raja Jurdak, and Ian Atkinson

- 74 **Moving Beyond the Turing Test**
Computers interacting with, not imitating, humans is the way forward.
By Robert M. French

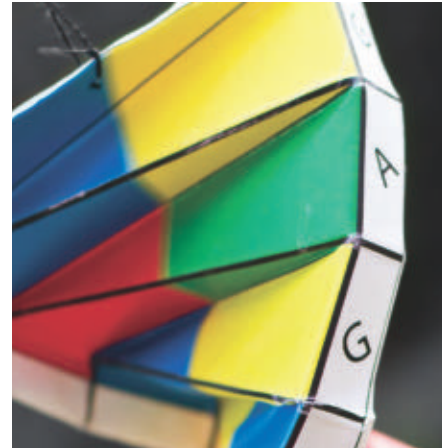


About the Cover:

The Turing Test has long been recognized as the true measure of machine intelligence. Will machines ever be able to answer enough questions to completely fool a human into thinking it is "human"? This month's cover story challenges that notion, declaring it is time to stop requiring computers to pass the Turing Test

and accept the computer for what it is—a sophisticated source of information.

Review Articles



- 78 **Theory of Algorithmic Self-Assembly**
The challenge of programming molecules to manipulate themselves.
By David Doty

Research Highlights

- 90 **Technical Perspective**
High-Level Data Structures
By Yannis Smaragdakis

- 91 **An Introduction to Data Representation Synthesis**
By Peter Hawkins, Martin Rinard, Alex Aiken, Mooly Sagiv, and Kathleen Fisher

- 100 **Technical Perspective**
Natural Algorithms in a Networked World
By Ali Jadbabaie

- 101 **Natural Algorithms and Influence Systems**
By Bernard Chazelle



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

Executive Director and CEO

John White

Deputy Executive Director and COO

Patricia Ryan

Director, Office of Information Systems

Wayne Graves

Director, Office of Financial Services

Russell Harris

Director, Office of SIG Services

Donna Cappel

Director, Office of Publications

Bernard Rous

Director, Office of Group Publishing

Scott E. Delman

ACM COUNCIL

President

Vinton G. Cerf

Vice-President

Alexander L. Wolf

Secretary/Treasurer

Vicki L. Hanson

Past President

Alain Chesnais

Chair, SGB Board

Erik Altman

Co-Chairs, Publications Board

Ronald Boisvert and Jack Davidson

Members-at-Large

Eric Allman; Ricardo Baeza-Yates;

Radia Perlman; Mary Lou Soffa;

Eugene Spafford

SGB Council Representatives

Brent Hailpern; Joseph Konstan;

Andrew Sears

BOARD CHAIRS

Education Board

Andrew McGettrick

Practitioners Board

Stephen Bourne

REGIONAL COUNCIL CHAIRS

ACM Europe Council

Fabrizio Gagliardi

ACM India Council

Anand S. Deshpande, PJ Narayanan

ACM China Council

Jianguang Sun

PUBLICATIONS BOARD

Co-Chairs

Ronald F. Boisvert; Jack Davidson

Board Members

Marie-Paule Cani; Nikil Dutt; Carol Hutchins;

Joseph A. Konstan; Ee-Peng Lim;

Catherine McGeoch; M. Tamer Ozsu;

Vincent Shen; Mary Lou Soffa

ACM U.S. Public Policy Office

Cameron Wilson, Director

1828 L Street, N.W., Suite 800

Washington, DC 20036 USA

T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association

Chris Stephenson,

Executive Director

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

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

STAFF

DIRECTOR OF GROUP PUBLISHING

Scott E. Delman

publisher@cacm.acm.org

Executive Editor

Diane Crawford

Managing Editor

Thomas E. Lambert

Senior Editor

Andrew Rosenbloom

Senior Editor/News

Jack Rosenberger

Web Editor

David Roman

Editorial Assistant

Zarina Strakhan

Rights and Permissions

Deborah Cotton

Art Director

Andrij Borys

Associate Art Director

Margaret Gray

Assistant Art Directors

Mia Angelica Balaquiot

Brian Greenberg

Production Manager

Lynn D'Addesio

Director of Media Sales

Jennifer Ruzicka

Public Relations Coordinator

Virginia Gold

Publications Assistant

Emily Williams

Columnists

Alok Aggarwal; Phillip G. Armour;

Martin Campbell-Kelly;

Michael Cusumano; Peter J. Denning;

Shane Greenstein; Mark Guzdial;

Peter Harsha; Leah Hoffmann;

Mari Sako; Pamela Samuelson;

Gene Spafford; Cameron Wilson

CONTACT POINTS

Copyright permission

permissions@cacm.acm.org

Calendar items

calendar@cacm.acm.org

Change of address

acmhlp@cacm.org

Letters to the Editor

letters@cacm.acm.org

WEB SITE

<http://cacm.acm.org>

AUTHOR GUIDELINES

<http://cacm.acm.org/guidelines>

ACM ADVERTISING DEPARTMENT

2 Penn Plaza, Suite 701, New York, NY

10121-0701

T (212) 626-0686

F (212) 869-0481

Director of Media Sales

Jennifer Ruzicka

jen.ruzicka@hq.acm.org

Media Kit acmmidasales.acm.org

Association for Computing Machinery (ACM)

2 Penn Plaza, Suite 701

New York, NY 10121-0701 USA

T (212) 869-7440; F (212) 869-0481

EDITORIAL BOARD

EDITOR-IN-CHIEF

Moshe Y. Vardi

eic@cacm.acm.org

NEWS

Co-Chairs

Marc Najork and Prabhakar Raghavan

Board Members

Hsiao-Wuen Hon; Mei Kobayashi;

William Pulleyblank; Rajeev Rastogi;

Jeannette Wing

VIEWPOINTS

Co-Chairs

Susanne E. Hambrusch; John Leslie King;

J Strother Moore

Board Members

P. Anandan; William Aspray; Stefan Bechtold;

Judith Bishop; Stuart I. Feldman;

Peter Freeman; Seymour Goodman;

Mark Guzdial; Richard Heeks;

Rachelle Hollander; Richard Ladner;

Susan Landau; Carlos Jose Pereira de Lucena;

Beng Chin Ooi; Loren Terveen

PRACTICE

Chair

Stephen Bourne

Board Members

Eric Allman; Charles Beeler; Bryan Cantrill;

Terry Coatta; Stuart Feldman; Benjamin Fried;

Pat Hanrahan; Tom Limoncelli;

Marshall Kirk McKusick; Erik Meijer;

George Neville-Neil; Theo Schlossnagle;

Jim Waldo

The Practice section of the CACM

Editorial Board also serves as

the Editorial Board of *queue*.

CONTRIBUTED ARTICLES

Co-Chairs

Al Aho and Georg Gottlob

Board Members

William Aiello; Robert Austin; Elisa Bertino;

Gilles Brassard; Kim Bruce; Alan Bundy;

Peter Buneman; Erran Carmel;

Andrew Chien; Peter Druschel; Carlo Ghezzi;

Carl Gutwin; James Larus; Igor Markov;

Gail C. Murphy; Shree Nayar; Bernhard

Nebel; Lionel M. Ni; Sriram Rajamani;

Marie-Christine Rousset; Avi Rubin;

Krishan Sabnani; Fred B. Schneider;

Abigail Sellen; Ron Shamir; Yoav Shoham;

Marc Snir; Larry Snyder; Manuela Veloso;

Michael Vitale; Wolfgang Wahlster;

Hannes Werthner; Andy Chi-Chih Yao

RESEARCH HIGHLIGHTS

Co-Chairs

Stuart J. Russell and Gregory Morrisett

Board Members

Martin Abadi; Sanjeev Arora; Dan Boneh;

Andrei Broder; Stuart K. Card; Jon Crowcroft;

Alon Halevy; Monika Henzinger;

Maurice Herlihy; Norm Jouppi;

Andrew B. Kahng; Xavier Leroy;

Mendel Rosenblum; Ronitt Rubinfeld;

David Salesin; Guy Steele, Jr.; David Wagner;

Alexander L. Wolf; Margaret H. Wright

WEB

Chair

James Landay

Board Members

Gene Golovchinsky; Marti Hearst;

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



ACM Copyright Notice

Copyright © 2012 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@acm.org or fax (212) 869-0481.

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

Subscriptions

An annual subscription cost is included in ACM member dues of \$99 (\$40 of which is allocated to a subscription to *Communications*); for students, cost is included in \$42 dues (\$20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is \$100.

ACM Media Advertising Policy

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

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhlp@cacm.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:10.1145/2380656.2380657

Mary Hall

Understanding ACM's Past

Computing is such a rapidly changing field—one that is always looking ahead to the next innovative technology—that little attention has been paid to looking back

toward its origins. ACM was founded in 1947 and now some 65 years later the founders of the field are largely gone and the early practitioners are aging. We know a lot about what technologies were invented, but far less about why or how. We can easily find factual information about the association's past, such as the date it was created or the officers of a Special Interest Group (SIG), but not the story behind why the SIG was formed. How do we gather such stories to share before it's too late?

These concerns prompted the creation of the ACM History Committee in 2004 by David Wise and Rick Snodgrass. The History Committee's charter is "to foster preservation and interpretation of the history of the ACM and its role in the development of computing" (see <http://history.acm.org>). The committee is composed of two overlapping groups: computer scientists and active representatives of ACM SIGs and committees who understand computing technology and the history of specific subfields; and computing historians and leaders of computing history organizations who understand the reasons for telling the story. This eclectic collection of expertise makes it possible for the committee to consider the many meaningful facets of ACM's rich history.

For a typical ACM member, the most visible contribution of the ACM History Committee is the recent launch of the Turing Award Web site, which announced this year's recipient Judea Pearl (see <http://amturing.acm.org>). In anticipation of the ACM A.M. Turing

Centenary Celebration last June, the History Committee worked with ACM Headquarters staff to plan a Web site worthy of what is frequently called the "Nobel Prize in Computing." The Curator-in-Chief, Mike Williams, Professor Emeritus at University of Calgary, led an intensive effort to identify and manage over 40 authors in order to populate this site in a year's time. Designed by the Ahlgrim Design Group, the site was brought to life with the help of ACM Headquarters staff Wayne Graves and Haritini Kanthou, who implemented a framework for integrating entries from numerous sources.

Another significant project of the committee is funding travel fellowships for historical study of ACM-related archives. We have supported both junior and senior researchers, computer scientists, and historians. Recent topics include Edmund Berkeley, one of ACM's founders; ACM computer graphics; European contributions to the Open Systems Interconnection (OSI) Reference Model; and ACM's pioneering computer science curriculums. We have also commissioned a number of oral histories of Turing Award recipients, early ACM presidents, staff, volunteers, and jour-

nal editors that can be found in the ACM Digital Archive and linked from our Web site.

If you are nearing retirement, you might be wondering what you should do with the correspondence, manuals, photos, and physical artifacts you have collected throughout your career. The items that are unique and of historical interest should be archived, but finding the pearls in an office full of boxes can be challenging. Our site offers guidelines on how to approach archiving your personal collection, which may help answer some of these questions.

Looking toward the future, a looming challenge in archiving computing history is that so much of the present is being created in the digital world, in email messages that use different mail programs, documents created by different software, and data in different formats. It may also be saved on media that has a short shelf life, for example, floppy disks from original PCs. Just being able to read these materials is technically challenging, but even more difficult is separating valuable information from the routine. How many email messages, even from Turing Awardees, are actually relevant to archiving computing history? This question of archiving electronic materials is one that will demand computing innovation not yet realized in order to examine and store our computing history. 

How do we gather such stories to share before it's too late?

Mary W. Hall (hallmw@acm.org) is a professor in the School of Computing at the University of Utah, Salt Lake City, and the chair of ACM's History Committee.

© 2012 ACM 0001-0782/12/12

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





DOI: 10.1145/2380656.2380658

Vinton G. Cerf

Computer Science Revisited

IN A RECENT column, I questioned whether there was any “science” in computer science. This provoked a great many responses that provided some very valuable perspective. To the degree that some aspects of computing are subject to analysis and modeling, it is fair to say there is a rigorous element of science in our field. Computability, analytical estimates of work to factor numbers, estimates of parsing complexity in languages, estimates of the precision of computations, computational costs of *NP*-complete problems, among others, fall into the category I call “analytic” in the sense we can say fairly strongly how computational cost grows with scale, for example. In recent conversations with Alan Kay, Edward Feigenbaum, Leonard Kleinrock, and Judea Pearl, I have come to believe that certain properties contribute to our ability to analyze and predict behaviors in software space.

Alan Kay spoke eloquently at a recent event honoring Judea Pearl’s ACM Turing Award. Among the points that Alan made was the observation that computing is not static. Rather it is a dynamic process and often is best characterized as interactions among processes, especially in a networked environment. The challenge in understanding computational processes is managing the explosive state space arising from the interaction of processes with inputs, outputs, and with each other.

In a recent discussion with Edward Feigenbaum, he noted that models used in hardware analysis are more tractable than those used in software analysis because what is possible in the “physical” world necessarily constrains the search for solutions. Software systems are designed and analyzed in “logical” space, in which there is far less structure to exploit in the analysis. However, the enormous

computing speeds possible today, combined with some structuring of the software design space, might allow deep and effective analyses of software as yet unavailable to mathematical methods or human thought. He recalled how very fast search, structured by some knowledge of chess, allowed IBM’s Deep Blue to defeat the world’s chess champion Kasparov. In one game, Deep Blue, exploring hundreds of millions of possibilities in the analysis of a move, found a brilliant solution path, probably never before seen, that led Kasparov to resign from the game and lose of the match.

Structure that constrains state spaces is sometimes conferred through abstraction. Here we get to another key observation about the potential for scientific approaches to computer science. To the degree that abstraction eliminates unimportant details and reveals structure, abstraction is a powerful analytical tool. It strikes me that Chaos theory illustrates this notion because patterns emerge in this theory despite the apparent randomness of the processes it seeks to analyze. If programs and algorithms are subject to suitable abstraction, it may be possible to model them with adequate but abstract fidelity so as to render the model analyzable. An example is found in queueing theory in which complex processes are modeled as networks of queues. The models are analytic under the right conditions. Kleinrock made enormous contributions to network design and analysis through the construction of queueing theoretic models. His most famous contribution was the recognition that the state space explosion could be tamed by his Independence assumption in which the statistics of the traffic flowing through the network were regenerated at each node using statistically identical but independent variables.

Modeling is a form of abstraction and is a powerful tool in the lexicon of computer science. This leads me to a final observation illustrated by Judea Pearl’s brilliant lecture on the use of Bayesian analysis to draw conclusions from problems involving probabilities rather than fixed values. Pearl’s theories of causal reasoning in conditional probabilities are often aided by graph-like models linking the various conditional statements in chains of cause and effect. The diagrams make it possible to construct analytic equations that characterize the problem and make the solution computable. It struck me that Pearl’s use of diagrams had an analogue in Richard Feynman’s diagrams of quantum interactions. These diagrams are abstractions of complex processes that aid our ability to analyze and make predictions about their behavior. Pearl’s diagrams may prove as powerful for science (not only computer science) as Feynman’s have for quantum physics and cosmology.

I have come away from this foray into computer ‘science’ with several conclusions. The first is there really is science to be found in computer science. The second is abstraction and modeling are key to making things analytic. The third is there is a lot of complex detail in computer programs and collections of interacting programs that has not admitted much in the way of abstraction or modeling, rendering these complex systems difficult to analyze. Finally, I believe our ability to understand and predict software behavior may rest in the invention of better high-level programming languages that allow details to be suppressed and models to emerge. I hope Alan Kay’s speculation will lead to serious improvements in the way we design and program computer systems.

Vinton G. Cerf, ACM PRESIDENT

Why Open Access?

CONCERNING THE PROPOSED Research Works Act, introduced in the U.S. Congress in 2011, Moshe Y. Vardi's "Editor's Letter" "Why ACM?" (Sept. 2012) left me a little confused as to Vardi's own position on both the law and open access. He wrote, "...most people who tell me that ACM should adopt open access seem to equate open access with free access, as in 'no one pays.'" Open access implies free access, but free access does not automatically imply some kind of free lunch or lead to a situation where no one pays. Taxpayers have already paid for the research and its access by funding the research.

Interpreting Vardi, the ACM position would seem to be: No implicit support for open access despite implicit support for the RWA by taking a neutral position. A respected professional scientific association like ACM should not take a neutral position here, as suggested by (past) ACM president Alain Chesnais in his President's blog earlier this year (<http://blog.acm.org/president/?p=67>).

At least two well-respected professional associations—the Optical Society of America (OSA) and the Institute of Physics (IOP)—do support open access, integrating the practice into their publishing policies and operations. Ignoring such support for open access would represent gross arrogance on the part of ACM; see the OSA and IOP positions spelled out at http://www.opticsinfobase.org/submit/review/pub_charge.cfm#OA and <http://iopscience.iop.org/info/page/openaccess>. OSA also supports the idea of sponsorship; for example, any existing article can be made open access to the general public regardless of its publication date by subsidizing the cost of publication, usually including for peer reviews, lifetime operation, and long-term maintenance of the article in the organization's digital library. So Vardi's argument that open access means a financial burden for ACM does not hold up entirely.

Dissemination of knowledge is a core ACM value involving access to knowledge and pursuit of scientific inquiry. ACM must weigh its position as a not-for-profit professional scientific association, not as a commercial publisher. I appreciate that publishing is costly, but in almost all cases I know the cost of publishing is covered through government research grants.

Open access to publicly funded scientific research in the public interest is fundamental to an open society and to ACM's own values. ACM should therefore move toward some kind of open-access publishing. Perhaps it can introduce optional open access under certain conditions following the OSA and IOP models.

Mehmet Süzen, Bonn, Germany

I was impressed by Moshe Y. Vardi's willingness to engage the ACM membership, including me (Lifetime Member since 2008), concerning open-access publishing (Sept. 2012), and by ACM for not being dogmatic about its copyright policy (such as by not requiring copyright transfer in all instances, like IEEE, and by introducing Author-Izer). I was also pleasantly surprised to see that publishing is the source of only one-third of ACM's annual revenue stream, which represents a great opportunity to develop a business model that would enable (much more free) access to (many more) people. ACM could easily get stuck in its current publishing model, blinding itself to potentially groundbreaking opportunities; for a sad example of such ostrich-like thinking, see the recording industry.

As a way to help ACM move in the right direction, please consider the following:

Opt-in. Provide an opt-in option for, say, a higher-price level of membership. In particular, some members whose subscription fee is paid by universities and employers or who are able to claim that fee on their tax returns might be willing to take such an option. Some current members might

be willing to pay more simply because they see the goal (and the organization) as worthwhile and worthy, the way some people donate to keep Wikipedia free;

Crowd-funding. Determine the amount ACM needs to publish all its articles and papers in a year, then ask ACM members, libraries, foundations, and other subscribers to contribute toward that cost—basically a form of crowd-funding. (Only if the amount is actually raised would ACM then make the articles and papers available.); and

Set free. Determine the amount needed to make a paywalled article available for free, amending it to each article and paper. If potential readers would be willing to pay that amount or make a micro-donation (such as via Flatter), then the amount would go toward the goal until the article or paper becomes freely available to all.

ACM could develop an innovative business model leading to totally free access, not as in "no one pays" but as in redistributing who pays.

Holger Kienle, Berlin, Germany

Author's Response:

Süzen seems to have misread my editorial. Please note ACM issued a clear statement against RWA: ACM supports open access in several ways, including allowing personal and institutional repositories and the Author-Izer service. Dissemination of knowledge is a key value of ACM, but as ACM members, we need to decide how to cover the cost of publishing; for example, both OSA and IOP allow authors to pay to open access to their articles. ACM is considering adopting this approach.

Moshe Y. Vardi, Editor-in-Chief

Not the Database World We Know

Communications readers have a right to expect accuracy. Sadly, accuracy is not always what they get. The article "All Your Database Are Belong to Us" by Erik Meijer (Sept. 2012) contains so many inaccuracies, confusions, and errors regarding "the database

world” it is difficult to read coherently. The first paragraphs alone contain more egregious misstatements than most entire articles or papers. For the record: “The raw physical data model” is categorically not “at the center of the [relational database] universe.” Queries do not “assume intimate details of the data representation (indexes, statistics, metadata).” While database technology relies on “The Closed World Assumption,” this assumption has nothing to do with what the author apparently meant. Every phrase in “Exposing naked data and relying on declarative magic becomes a liability” relies on at least one counterfactual. “Objects should hide their private data representation, exposing it only via well-defined behavioral interfaces.” But this is exactly what the relational model does—except (unlike OO) it adopts an interface discipline that makes ad hoc query and the like possible. “In the realm of [data] modelers, there is no notion of data abstraction.” Astonishingly wrong. “[Database technology necessarily involves] a computational model with a limited set of operations.” False. Although the (very powerful, well-defined, provably correct) required set of relational operations is small, the sky’s the limit on derived relational operations or operations that define abstract data type/domain behavior.

The author’s unfounded antipathy toward relational databases dominates even his application of CAP: “The problem with SQL databases... is the assumption that the data... meets a bunch of consistency constraints that is difficult to maintain in an open [‘anything goes?’] distributed world.” CAP does not eliminate this requirement; “...the hidden cost of forfeiting [system-enforced] consistency... is the need [for the programmer] to know the system’s invariants.”¹ Nor can programmers “...design their systems to be robust...to inconsistency.” Once data inconsistency invades a computationally complete system, it is not even, in general, detectable, and all bets are off. Consistency must be enforced, hence constraints. The author seemed to equate detecting abnormal execution with enforcing logical data consistency. No wonder

confusion abounds; CAP consistency is single-copy consistency, a subset of what ACID databases provide, yet the Gilbert/Lynch CAP proof relies on linearizability, a more stringent requirement than the serializability ACID databases need or use.

And so on... Deconstructing the entire article properly would take more time than we care to devote, but the foregoing should suffice to demonstrate its fallaciousness. We hope the author is not teaching these confusions, errors, logical inconsistencies, and fallacies.

It is difficult even to believe the article was peer reviewed. Indeed, it is truly distressing it did not demonstrate even minimal understanding of one of the most important contributions to computing: the relational model. We can only deplore *Communications*’s role in promulgating such a lack of understanding.

C.J. Date, Healdsburg, CA,
and **D. McGovern**, Boulder Creek, CA

Reference

1. Brewer, E. CAP twelve years later: How the ‘rules’ have changed. *IEEE Computer* 45, 2 (Feb. 2012), 23–29.

Author’s Response:

The purpose of the article was not to criticize the relational model but to point out how building industrial-strength systems using today’s relational database systems requires leaving the ivory tower and dealing with a morass of ad hoc extensions to the clean mathematical basis of first-order predicate logic. Rather than depend on pure sets and relations, developers need to think in terms of (un)ordered multisets. For the sake of efficiency and lock-contention avoidance, transactions allow for various isolation levels that clearly violate the ACID guarantees of Platonic transactions. The article also considered whether in the new world of the Cloud we should view as complementary computational models that fundamentally address loosely coupled distributed systems, like Carl Hewitt’s Actors.

Erk Meijer, Delft, The Netherlands

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.

© 2012 ACM 0001-0782/12/12

Coming Next Month in COMMUNICATIONS

**Human Mobility
Characterization from
Cellular Network Data**

**Abstractions
for Genomics**

**Computer Security
and the Modern Home**

**What College
Could Be Like**

**Conference-Journal
Hybrids**

**Browser Security:
Appearances
Can Be Deceiving**

**The Web Won’t
Be Safe or Secure
Until We Break It**

**Condos
and Clouds**

**How Mechanical
Assemblies Work**

**Plus the latest news about
side-channel attacks,
beyond Hadoop, and
stealing others’ content.**



Association for Computing Machinery

Global Reach for Global Opportunities in Computing



Dear Colleague,

Today's computing professionals are at the forefront of the technologies that drive innovation across diverse disciplines and international boundaries with increasing speed. In this environment, ACM offers advantages to computing researchers, practitioners, educators and students who are committed to self-improvement and success in their chosen fields.

ACM members benefit from a broad spectrum of state-of-the-art resources. From Special Interest Group conferences to world-class publications and peer-reviewed journals, from online lifelong learning resources to mentoring opportunities, from recognition programs to leadership opportunities, ACM helps computing professionals stay connected with academic research, emerging trends, and the technology trailblazers who are leading the way. These benefits include:

Timely access to relevant information

- *Communications of the ACM* magazine
- *ACM Queue* website for practitioners
- Option to subscribe to the *ACM Digital Library*
- ACM's **50+ journals and magazines** at member-only rates
- *TechNews*, tri-weekly email digest
- *ACM SIG conference* proceedings and discounts

Resources to enhance your career

- **ACM Tech Packs**, exclusive annotated reading lists compiled by experts
- **Learning Center** books, courses, podcasts and resources for lifelong learning
- Option to join **37 Special Interest Groups (SIGs)** and **hundreds of local chapters**
- **ACM Career & Job Center** for career-enhancing benefits
- *CareerNews*, email digest
- **Recognition of achievement** through Fellows and Distinguished Member Programs

As an ACM member, you gain access to ACM's worldwide network of more than 100,000 members from nearly 200 countries. ACM's global reach includes councils in Europe, India, and China to expand high-quality member activities and initiatives. By participating in ACM's multi-faceted global resources, you have the opportunity to develop friendships and relationships with colleagues and mentors that can advance your knowledge and skills in unforeseen ways.

ACM welcomes computing professionals and students from all backgrounds, interests, and pursuits. Please take a moment to consider the value of an ACM membership for your career and for your future in the dynamic computing profession.

Sincerely,

Vint Cerf

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: AD13

You can join ACM in several easy ways:

Online

<http://www.acm.org/join>

Phone

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

Fax

+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/about/code-of-ethics>

choose one membership option:

PROFESSIONAL MEMBERSHIP:

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

STUDENT MEMBERSHIP:

- ☐ ACM Student Membership: \$19 USD
- ☐ ACM Student Membership plus the ACM Digital Library: \$42 USD
- ☐ ACM Student Membership PLUS Print CACM Magazine: \$42 USD
- ☐ 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.

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

☐ Professional Member Dues (\$99 or \$198) \$ _____

☐ ACM Digital Library (\$99) \$ _____

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

Total Amount Due \$ _____

Card # _____

Expiration date _____

Signature _____

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



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

DOI:10.1145/2380656.2380660

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

Levels of Abstraction; Pre-Teens and Career Choices

Mark Guzdial writes about the need for programming languages to support multimedia at all levels.

Judy Robertson shares insights about 12-year-old students' lack of understanding about computer science.



Mark Guzdial
"Why Don't Languages Support Multimedia All the Way Down?"

<http://cacm.acm.org/blogs/blog-cacm/109917>
June 22, 2011

Donald Knuth gave the keynote talk at ITICSE 2003 on "Bottom-Up Education." He argued that the hallmark of thinking like a computer scientist was being able to shift levels of abstraction, from the highest levels of application, all the way down to the bits, if necessary. He was arguing for his MMIX processor, but the same argument can be made in lots of different pedagogical contexts.

That's really what Barbara Ericson and I are doing in our Media Computation approach to introductory computing. Students today use digital media every day. They recognize the manipulation of media as a relevant and useful activity. In our approach, we teach students to manipulate digital sounds at

the sample level and digital pictures at the pixel level. They can then write simple loops to create Photoshop-like effects, like flipping an image or removing red eye, or to create digital sound effects, like creating an echo, splicing, or reversing sounds. Manipulating pixels and samples is fun and easy—we've shown that it's a CS1-level activity. It's another case of manipulating the lowest-levels of abstraction to create an effect at the application level.

The problem is finding languages and libraries that support this level of access and manipulation. Sure, lots of languages can show pictures and play sounds, but that's getting stuck at Knuth's highest level of abstraction. How many languages and libraries, even those aimed at students, let you shift levels of abstraction with media?

Barbara and I wrote our books in Python and Java by cheating. Java does support shifting levels of abstraction. We chose a version of Python, Jython, that lets us reuse the classes we wrote

in Java. I have also been able to construct our Media Computation examples in Squeak as well. Jennifer Burg has shown how easily it can be done in C. Then that's really about it.

Our publisher has encouraged us to look into using Media Computation with other languages, especially Python 3.0. And that's where we run into problems. We can manipulate pixels; in fact, Nick Parlante at Stanford has started teaching JavaScript using Media Computation with pixel-level manipulations. A recent review of audio packages for Python shows that *none* of them support sample-level manipulations cross-platform. I have been able to write small examples in PyGame, but there are some significant bugs in that package. For example, if you open up a sound that is not CD quality, PyGame "re-samples" the sound, so a sound that you open and save back out might double in size. If you care about the byte level, it is disconcerting for more of them to appear without warning.

I have found no packages that let me do pixel- and sample-level manipulations in other languages. There is a book on learning Haskell with multimedia, but it is all at the highest level of abstraction. I have tried to find such supports for Scheme, but the only audio package I have found allows you to play sounds, but you still can't access the samples in those sounds. It's frustrating because, if a language or library supports *playing* the sounds, then those samples are *somewhere* there in memory. Let us at them!

Now, I'll bet there are libraries for

manipulating pixels of images and samples of sounds in many of these languages, but my experience suggests that they're not obvious, not easy to find. Why not? Don't we think that Donald Knuth is right, and it really is important for CS students to be able to get all the way down *easily* and *obviously* to understand how to build it all back up?

There is an argument that real application developers don't typically work at that level. Video game programmers leave the pixel and sample manipulations to the gaming engine. Most application developers just want to show pictures and play sounds and videos. But that doesn't excuse not providing access for *students*. Learning is a conscious process. It's so much easier to be conscious about things we can see. How do you study something that you can't see, that you can't manipulate? How do you learn samples and pixels if they're always hidden inside some library or engine? Sure, it's possible to learn things that are invisible, but it works much better if they are visible, accessible, and malleable.

Media is something that I care about, but I wonder if it's an instance of a larger problem. It's important for students to shift levels of abstraction. How well do our languages for students support shifting levels of abstraction; that is, being able to see everything from the application level down to the bytes? And if they don't, we should be asking, "Why not?"



Judy Robertson
"Foggy Futures: The Confused Computing Career Aspirations of 12-Year-Olds"

<http://cacm.acm.org/blogs/blog-cacm/115085>
 July 25, 2011

Can you remember what it was like to be 12 years old and have an infinity of possible careers in front of you? What made you choose computing? Was it a positive choice, or did you drift into it? I have been thinking about this today because I have been listening to recordings of interviews with 12-year-old boys and girls about their attitudes to computing, and their future career choices.

I chose computing because it was difficult. I wanted the challenge. I distinctly remember trying to work out

JUDY ROBERTSON

"I chose computing because it was difficult."

how to write a sorting algorithm as I trudged along my morning newspaper delivery route. Naturally because it was so hard it seemed the obvious thing to want to do with my life. (Go figure!) Back in those days, computers weren't part of everyday life. My exposure to computing was from learning to program at school, and from watching my dad type expert systems code from the back of a book into an Amstrad word processor.

But now, children's exposure to computing is ubiquitous and centered around the use of computers rather than more fundamental computer science concepts. In our recent interviews with 12-year-olds who had just completed a game-making project, we asked them about what they understood by the term "computing." It became clear that their understandings were partly related to the label for the subject on the timetable, such as "ICT," "Information Technology," or "Computing Studies." None of the classes were labeled "Computer Science."

When asked what they might expect to do in a computing class, the children typically told us about using applications: spreadsheets, databases, PowerPoint, Word, and sound recording packages. The "Internet" was often featured, in the sense of learning to use Internet-based applications safely and effectively. They thought that in a computing class they might learn how to use computers in general, and learn to use programs they had not come across before. A couple of students mentioned learning about what computers can do, and what parts are inside them. Oddly, no one mentioned that they would expect to study the fundamental properties of computation, or the patterns for effective software design.

In terms of future careers, the students often explained that while they thought computing was an important aspect of many lines of work, it was not

something they wished to focus on. A boy who wanted to be a pilot mentioned that "there are a lot of computers in that. You have to login when you're going out and log out and your computer's inside the plane." A girl who wanted to be a doctor conceded that she would learn computing if it were necessary to do the job. Worryingly, a couple of the girls had misconceptions about how programming might fit into careers:

Girl A: "To be an optician or a vet, you have to use the computer quite a lot for that."

Girl B: "Programming and stuff."

Girl A: "To be an optician you have to program what it is, know what it is, certain parts. Like what's wrong, how they can help and stuff."

Interviewer: "Have you done any programming yet in school?"

Girl A: "I don't know."

Girl B: "We did. We did our own program. 'My computer of the future,' that was a programming project."

Girl A: "We know that programming is like typing and stuff."

Girl B: "Is it?"

Girl A: "So I believe...."

Typing? Opticians? This calls into question an attitude questionnaire I recently used that included a perfectly reasonable-seeming question about how much the respondent enjoyed programming. The results may not be very reliable if some of the kids think programming is merely typing.

This brings me to a broader point about computer science education. With many excellent initiatives to encourage students to study computing under way, we are going to need to evaluate their effectiveness. To do so, we need surveys that reliably and validly uncover changes in attitudes to computing. But such instruments will need to be designed very carefully if there is such a mismatch between researchers' and students' understanding of basic terms such as "computing" and "programming." Perhaps vocabulary development needs to be part of the computer science education itself. We need to clearly articulate to pre-teens what computer science is, as well as why it is so important. 

Mark Guzdial is a professor at the Georgia Institute of Technology. Judy Robertson is a lecturer at Heriot-Watt University.

© 2012 ACM 0001-0782/12/12

ACM's Career & Job Center

Looking for your next IT job?

Need Career Advice?

Visit ACM's Career & Job Center at:

<http://jobs.acm.org>

Offering a host of career-enhancing benefits:

- A highly targeted focus on job opportunities in the computing industry
- Access to hundreds of corporate job postings
- Resume posting keeping you connected to the employment market while letting you maintain full control over your confidential information
- An advanced Job Alert system notifies you of new opportunities matching your criteria
- Career coaching and guidance from trained experts dedicated to your success
- A content library of the best career articles compiled from hundreds of sources, and much more!

The ACM Career & Job Center is the perfect place to begin searching for your next employment opportunity!

<http://jobs.acm.org>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

ACM Member News

SEARCHING FOR BETTER SOLUTIONS



Norbert Fuhr discusses his work on information retrieval (IR) almost like a professor

explaining an incomplete grade on a student's term paper: There's still much left to be done.

"I think most users are quite satisfied with the quality of IR they're experiencing, at least for routine tasks," says Fuhr, leader of the Duisburg Information Engineering Group at the University of Duisburg-Essen. "But IR is not an engineering science yet. We are not able to make predictions for new applications involving new document collections, search-task types, or relevance criteria. Many organizations have such applications, but only for a restricted user community. We should collect empirical evidence for a broader range of collections, information needs, and notions of relevance."

Fuhr is widely considered one of the leading pioneers in research to benefit the search industry, making significant advances in the development of probabilistic retrieval models for databases and XML. Of special note is his research on models that paved the way for ranked approaches in search operations. For these and other contributions, ACM SIGIR honored Fuhr with the 2012 Gerard Salton Award.

Fuhr is now focusing on context and customization. System functionality should support a wide range of information-access needs, he says, and the interface experiences should be tailored for different user groups and task categories. "For a current project in the medical domain, for example, we're developing different interfaces for laymen, medical practitioners, and radiologists. My grand vision is about systems being able to manage all kinds of information—facts, texts, or multimedia—with these systems being aware of the user's current work task and then providing the necessary information."

—Dennis McCafferty

CREDIT: TK

Quantum Quests

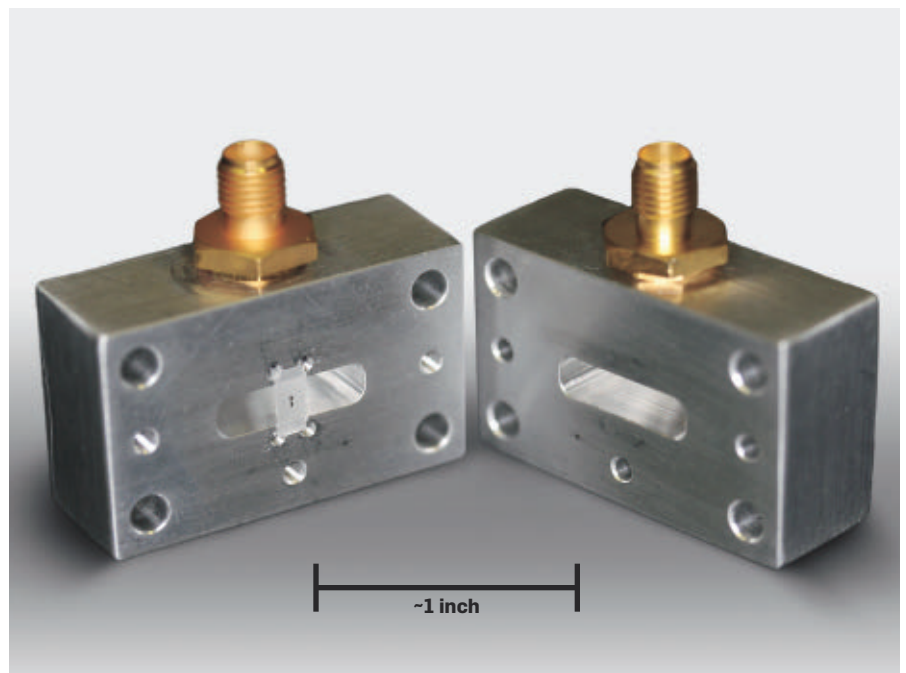
Three breakthrough experiments involving photons have extended coherence times and indicated scalable production.

HARNESSING THE PROPERTIES of photons is assumed to be one of the fundamental routes to the creation of viable quantum computing architectures. Those architectures will, among other features, need to maintain coherence, or a stable quantum bit (qubit) state, long enough for the algorithm to complete in the coherence time, which may require error correction to enable such completion and reliability; a fabrication process that allows carryover from today's mass manufacturing process of integrated circuits; and nodes that are identical in construction and capabilities in order to scale successfully.

Recent experimental results in harnessing or stabilizing photons from IBM's research facility in New York; the University of Waterloo, the University of Toronto, and the University of Innsbruck; and the Max Planck Institute of Quantum Optics have addressed these fundamental conditions in a promising fashion. The methods used in the experiments offer the promise of possible future applications for a quantum Internet, or quantum network, and quantum key distribution.

Cavities and Coherence Times

Two of the projects—an IBM experiment examining superconducting qubit



IBM's superconducting qubit device contains a qubit, about 1mm in length, suspended in the center of the cavity on a Sapphire chip. The cavity is formed by closing the two halves.

bits and the Max Planck experiment examining how best to preserve quantum state during transport—yielded a similar finding although examining very disparate areas: Coherence times and control over quantum elements were improved by employing cavities to generate or preserve essential qualities of those elements.

The IBM researchers demonstrat-

ed they could maintain a stable qubit state approaching 100 microseconds—an interval they claim falls well within the range of coherence required for large-scale tests of error correction and fault-tolerant quantum computing procedures—by stifling the amount of photons and the actions of those photons near a three-dimensional superconducting qubit. The qubit is about

one millimeter in length, suspended in a super-cooled copper cavity about 38 millimeters wide.

The IBM work is significant, according to published commentary by Yale University physics professor Steven Girvin, because it is the most recent example of work that has improved the coherence time of superconducting qubits by five orders of magnitude in just over a decade. In his commentary, Girvin contends that “imperfect microwave hygiene,” rather than any intrinsic materials limits, has been the aggravating factor in limiting coherence times. Girvin says continuing experimentation is demonstrating that residual qubit dephasing, or reversion to classical state, is being caused by stray photons entering the cavities surrounding the superconducting qubits.

According to Matthias Steffen, manager of experimental quantum computing at IBM’s Thomas J. Watson Research Center, protection against dephasing was achieved by equalizing the temperature of photons within the cavity with that of the walls of the cavity itself.

“During the qubit operation the cavity should ideally contain no photons or, strictly speaking, the photon number in the cavity should not fluctuate,” Steffen says. “In our case, we found the cavity population does fluctuate when the cavity is not properly heat sunk, and when it is overcoupled to signal lines which, in turn, are also not heat sunk well. Our solution then included a copper cavity so that the cavity is thermalized, making sure that no photons are present in the cavity during the qubit operation.”

The Max Planck researchers also employed a cavity in which to maximize their efforts. In this case, the creation of a single photon capable of storing the quantum information in a single atom and transferring it via optical fiber to an identical atom located 20 meters from the first. Max Planck researcher Stephan Ritter says the team chose a single gaseous rubidium atom as each node because it was easier to protect the atoms from the surrounding environment and because the energy levels within each atom are well defined, thereby making atom manipulation via laser possible.

“We produce the photon by shining the laser onto the atom from the side,”

Ritter says. “By precisely controlling the intensity of the light, you can control the creation of the single photon. The photon is deposited into the cavity, then leaks out. If you just shine the laser beam onto the atom, it can also emit a photon but that might go off in any direction. You want one directed channel into which it goes, and basically the cavity provides us with this clearly defined direction. We have tried to make each of the nodes the same; you can easily scale it up by adding more nodes.”

The cavity in which the photon is deposited is about 500 micrometers long, Ritter says, while the mirrors on either side are slightly larger, about 7.5 millimeters in diameter. The mirrors do not share symmetric reflecting properties, however. One mirror has a much higher transmission quotient, so that the resulting photon has what Ritter says is a 90% probability of leaking out on one side, allowing an operator to more easily couple the photon with the networked optical fiber.

The classical physics view of the action inside the cavity holds that the photon is reflected thousands of times in the effort to impart the atomic quantum state onto it. More accurately, Ritter says, the cavity enhances the interaction between the light field inside the cavity and the atom, leading to enhanced coupling of the atom and photon as compared to free space.

The overall success rate of quantum entanglement between two atoms in

The IBM research is significant because it has improved the coherence time of superconducting qubits by five orders of magnitude in just over a decade, notes Yale physicist Steven Girvin.

the experiment was 2%. Ritter says this is an improvement of many orders of magnitude over predecessor experiments working with ion entanglement in free space, although the closer one gets to the ultimate goal of 100% the more difficult it gets.

“What’s important is that the process gives us a handle on all the properties that such a single photon can have,” Ritter says. “We can control the frequency, the direction of the photon using the cavity, and the left- or right-hand polarization of the photon. This is really important because the polarization is what we encode the quantum information in.”

Photons from a Chip

The third experiment in the recent photonic trifecta, the demonstration of a method to produce entangled photon pairs on an integrated circuit, earns praise from Ritter as yet another key step in advancing quantum research further into the mainstream.

“They’ve done great work on integration, on making things small, on using existing microfabrication technology to produce something which might in the future be very useful for doing quantum information processing,” Ritter says. “Each of our nodes is only a single atom with a cavity around, but if you take into account all the optics and all the lasers, one of these nodes fills a whole lab at the moment.”

The Canadian and Austrian research team designed a photonic channel called a Bragg reflection waveguide on an aluminum-gallium arsenide chip that will allow laser light pumped into the chip to produce lower-energy entangled photons that exit the other side of the waveguide. The structure is likened to a many-layered cake by University of Toronto professor Amr Helmy, the chip’s designer. The “cake” consists of a core, surrounded by reflector layers, that will allow a desired wavelength of light to pass through, which results in the creation of the entangled photon pair. The wavelength can be tuned via altering the ratio of aluminum in each layer of the reflectors, as well as the thickness of each layer. This tuning allows the energy levels of the laser light and the photons to be in sync, or phase-matched, which enhances efficiency of the process.

The next step in the process is to integrate the laser source into the waveguide itself, according to team member Rolf Horn, a researcher at Waterloo. That will be achieved by constructing small structures called quantum wells, which will trap electrons supplied from a current source. As current is injected into the structure, these electrons will recombine with holes in the structure's valance band and emit light.

"At that critical current density, the loss in that mode will be less than the gain and you will have lasing," Horn says. "Your structure won't change at all, in the sense you'll have one monolithic source, where you don't have to couple a laser from some other component or piece of equipment into the waveguide. The structure will act as both a laser and an entangled photon pair producer, so it's all integrated into one package."

Thus far, the team has proved separately that the lasing process and the photon pair generation are viable, says Horn. The team expects to be able to tune the fabrication parameters to better couple the lasing wavelength and photon properties. Helmy says successful demonstration of the integrated technology may have widespread ramifications for quantum communication experiments.

"Instead of doing them in the lab and having to align things with mirrors and such, we can do all that with nanolithography," he says. "So, we can define it in a CAD program that a computer scientist can use. So, you're really bringing that stuff to the masses out of the lab."

Martin Laforest, manager of scientific outreach at the Institute for Quantum Computing at the University of Waterloo, says the experiments all indicate significant breakthroughs.

"Those are very encouraging results, all of them," Laforest says. "The IBM experiment is the sort that is really closing the gap between practicality and theory—same thing with the work at Max Planck. The idea of light-matter interaction has been around for a while, but now they've really been able to transfer from one quantum register to another 20 meters apart, in a 60-meter fiber. That's a massive result. The idea of quantum networks has been around for ages. Now it's a reality."

As for the work done in Waterloo, Toronto, and Innsbruck, Laforest says

"Those are very encouraging results, all of them," says Martin Laforest. "The IBM experiment is the sort that is really closing the gap between practicality and theory—same thing with the work at Max Planck."

successfully employing established fabrication techniques may revolutionize applications such as quantum key distribution or secure communications. With integrated laser and photon creation on a single chip, he says such technologies could be put on mobile devices, "and the idea of personal quantum cryptography is not a crazy idea." **C**

Further Reading

Duan, L.-M. and Monroe, C. Colloquium: Quantum networks with trapped ions, *Reviews of Modern Physics* 82, 2, April 28, 2010.

IBM Labs
IBM Research Announces New Advances in Quantum Computing, http://www.youtube.com/watch?v=_NRmOe1b8_s, Feb. 23, 2012.

Paik, H., et al.
Observation of high coherence in Josephson junction qubits measured in a three-dimensional circuit QED architecture, *Physical Review Letters* 107, 240501, Dec. 5, 2011.

Rigetti, C., et al.
Giant improvements in coherence time for superconducting qubits, *Journal Club for Condensed Matter Physics*, March 28, 2012.

Svozilik, J., Hendrych, M., and Torres, J.P.
Bragg reflection waveguide as a source of wavelength-multiplexed polarization-entangled photon pairs, *Optics Express* 20, 14, June 20, 2012.

Gregory Goth is an Oakville, CT-based writer who specializes in science and technology.

© 2012 ACM 0001-0782/12/12

Milestones

CS Awards

The Association for the Advancement of Artificial Intelligence (AAAI) and the Simons Foundation recently honored 11 computer scientists.

AAAI FELLOWS

AAAI recognized five computer scientists for their outstanding contributions to the theory or practice of AI by electing them AAAI Fellows. They are Murray S. Campbell, IBM T.J. Watson Research Center, for "significant contributions to computer game-playing, especially chess, and the associated improvement in public awareness of the AI endeavor"; Boi V. Faltings, Swiss Federal Institute of Technology, for "significant contributions to research and applications of spatial reasoning, distributed constraint reasoning, case-based reasoning, design, optimized communication, and social networking"; Yolanda Gil, University of Southern California, for "significant contributions to intelligent user interfaces, knowledge capture, representations for Web-based science, and for leadership in the AI community"; Peter F. Patel-Schneider, for "significant contributions to knowledge representation and reasoning, in particular the exploration of decidable logic fragments, reasoning optimizations, and the application of KR on the Web"; Francesca Rossi, Università degli Studi di Padova, for "significant contributions to constraint and preference reasoning, and for distinguished leadership and service"; and Peter H. Stone, The University of Texas at Austin, for "significant contributions to machine learning, multiagent systems, and robotics, and pioneering applications in the robot soccer, trading agents, and autonomous driving domains."

SIMONS INVESTIGATORS

The Simons Foundation selected five computer scientists as Simons Investigators in 2012, the inaugural year of the program. They are Sanjeev Arora, Princeton University; Shafira Goldwasser, Massachusetts Institute of Technology; Russell Impagliazzo, University of California, San Diego; Jon Kleinberg, Cornell University; and Daniel Spielman, Yale University.

—Jack Rosenberger

Zoom In, Zoom Out

Drilling down to more detail on a computer screen, or moving out to see the context, is basic. But it's hardly simple and, after 20 years, innovations are still occurring.

ZOOMING, THE ABILITY to move in and out of information to show different levels of detail on a computer display, would seem the most mundane of capabilities. Indeed, zooming in various guises has been possible for at least two decades, and it's the rare application today that does not give users at least a rudimentary zooming ability.

But the story of zoom technology is surprisingly complex. There are many ways to produce a zoom effect, and many of the ideas tried over the years have seemed to hold enormous promise, but have languished, often because the cost to the user exceeds the benefit that the user perceives.

Today innovation lies less in the development of new zoom technologies and more in the application of existing ones. One such application, called ChronoZoom, can seamlessly zoom into a gigantic multimedia database at levels from millennia to nanoseconds. This kind of deep zooming is an example of what has become known as the “zoomable user interface,” or ZUI.

It seems an impossibly ambitious goal: “visualizing the history of everything.” ChronoZoom, however, may plausibly enable just that. ChronoZoom is essentially an online interactive timeline using a nonlinear, or variable scale, presentation. At the top of your screen lies a horizontal timeline stretching at the left from the Big Bang 13.7 billion years ago to today on the far right. In between, you might zoom in on a point at 12 billion years ago and watch a video about how dying stars began to create new elements. Or you might zoom in to the year 1300 to learn about the birth of the Ottoman Empire, or to Jan. 24, 2012 to learn about the birth of Princess Athena Marguerite Françoise Marie of Denmark. At any point on this “virtual canvas” you could be presented with relevant photographs, graphs, video, audio,



A Web-based timeline, ChronoZoom enables users to see the scale of time over cosmic, geologic, biological, and social periods.

or text. “It could scale down to one nanosecond if we had content for a nanosecond,” says Rane Johnson-Stempson, a director at Microsoft Research.

ChronoZoom is an open source system led by the Outercurve Foundation and being built by scientists at the University of California, Berkeley, Microsoft Research, Moscow State University, the University of Washington, and nearly 20 subject matter experts from around the world. ChronoZoom runs on Microsoft’s Azure cloud service and uses SQL Azure for storage. It has HTML5 on the front end and some 100 APIs to integrate existing content and software on the back end. ChronoZoom also operates on a variety of platforms, including mobile devices such as iPads, iPhones, Android devices, and Windows Phone.

The problem ChronoZoom and other ZUIs attempt to solve could not be simpler. “There is more information than fits on the screen,” says Ben Bederson, a computer science professor at the University of Maryland and a pioneer in ZUI technology. This problem can be overcome by scrolling, linking, searching, and zooming into successively dense representations of the information. All file and Web browsers and image and document editors offer one or more of these capabilities.

But, according to Bederson, the term ZUI has come to refer specifically to those systems—such as Google Maps—that provide both spatially and temporally changing views to navigate at different scales, among multiple objects such as text documents, photographs, or entire ensembles such as presentations.

People are attracted to ZUIs for at least three reasons, Bederson says. They are engaging, naturally suited to human visual perception abilities; they are visually rich, offering multiple degrees of freedom; and they are potentially simple, offering the ability to find something in a place, much as you might locate a document on your physical desktop.

The March of ZUIs

Ken Perlin and David Fox, computer scientists at New York University, kicked off the ZUI era in 1993 with a paper titled “Pad—An Alternative Approach to the Computer Interface.” Pad was “an infinite, two-dimensional information plane that is shared among users,” they noted. “Every object occupies a well-defined region on the Pad surface.” Users peered into virtual magnifying glasses called portals, which could be moved to any part of the surface the user wanted to look into. Each mouse click instantly produced a view twice or half the size of

the previous view.

ZUI technologies appeared in a steady stream after that. A year later, Bederson and James Hollan, then at Bellcore, unveiled Pad++, a successor to Pad that, instead of sudden jumps, would zoom smoothly and continuously when the mouse button was held down. Ensuing ZUIs included CounterPoint, which Bederson introduced in 2001, a Microsoft PowerPoint plug-in that enabled presenters to arrange their slides on a zoomable space. In 2008, a company called Zoomorama introduced a client/server ZUI for zooming images over a network. And in 2009, Microsoft brought out Canvas for OneNote, which offered a zoomable overview of all of a user's notes.

A particularly dazzling ZUI appeared in 2009 in the form of Prezi, a “zooming presentation software,” from a Budapest-based company of the same name. Prezi is Web based and features 3D fade-in animation that combines smooth rotations, panning, and zooming in realistic 3D views of slide presentations. Prezi uses Adobe Flash in the browser and the desktop application, and uses native OpenGL, the cross-platform API for 2D and 3D graphics, on the iPad.

While each advance has been based on plausible expectations of user benefits, many ZUIs have enjoyed limited success in the marketplace. Zoomorama closed for lack of business in 2010. And PhotoMesa, a zoomable photo browser developed by Bederson in 2001 that simultaneously offered views of multiple directories of images, did not catch on either. “I used all kinds of tricks to help you organize your images, but in the end it was not a good idea,” explains Bederson. “It has this essential problem that its goal was to spatially organize tens of thousands of images. Nobody wants to do that.”

PhotoMesa was not the only ZUI to make that mistake, according to Bederson. These ZUIs are based on the observation that people often organize their lives spatially, such as papers arranged on their desks, for example. “But the reality is people don't have 10,000 papers on their desks,” says Bederson, who notes the problem is that people's spatial memories do not scale up. “You can't remember the position of 10,000 things. For these kinds of problems, you can't beat visual scanning of a one-

dimensional list.”

PhotoMesa users found they preferred to use simple programs like Microsoft Windows Explorer or Google Picasa, where files and folders are displayed linearly and sorted alphabetically or by date.

Of Prezi, Bederson says, “It's cool. The first time you see it you say, ‘Wow, this will change everything.’ But there is little real benefit and a real cost.” The cost, he says, comes from the need to sit through animated transitions, each of which takes a small amount of time. “It's a lot of distraction that ultimately annoys people,” he says.

But Prezi argues that its mass-market appeal—it claims to have 12 million users—proves its value. “Any technology misused can be distracting—disjointed video, poorly designed newsletters or intranets, and long wordy emails,” says Drew Banks, head of marketing at Prezi. “In general, research tells us that the human brain is far more adept at spatial memory than it is at remembering nonvisual or non-spatially related data.”

In any case, ZUIs have proven themselves, says Bederson, and the right approach now for application developers is to choose wisely among existing zoom technologies and apply them correctly. “This was a computer science research topic, with lots of people looking at lots of technologies,” says Bederson. “Some of them panned out and some didn't. I'm delighted to see zooming as part of the developer's toolkit, but I don't see it as a research project anymore.”

Perhaps not, but the use of deep zooming technology seems likely to increase as applications access ever larger and more heterogeneous databases. “The reason we use the ZUI metaphor is that you don't know what you are searching for unless you know,” says Johnson-Stempson. “When you have crazy amounts of data, it's really difficult. The whole point of ChronoZoom is to come across information you have no idea about, then zoom into the details.” ChronoZoom includes a rudimentary keyword search engine at present, but developers see zooming as the primary way for users to find information.

“In the field of big history, it is the patterns, cycles, and trends that are most useful,” says Roland Saekow, a ChronoZoom project leader at the

University of California, Berkeley. “By visualizing all the past, it is possible to browse history, rather than digging it out piece by piece. ZUIs excel at the capability of providing context.”

“I think one of the really interesting questions now is how to let the user feel more in control as they navigate these ZUIs,” says Saekow. “In the latest build, we have added multitouch support in HTML5. With trackpads and multitouch screens, we find it is less and less necessary to teach users how to zoom in or out. Pinch and zoom on such new surfaces are natural motions. Instead of having to learn that the scroll wheel controls zoom depth, and the mouse controls panning and selection, touch makes it possible to do both at once.

“Ultra-high resolution touch screen displays may make navigation of large-scale ZUIs more intuitive in the future,” says Saekow. “Perhaps with the addition of haptic feedback, we can provide users a sense of how far they are zooming, or as they zoom past certain depths, we can provide a tactile feedback. We have also not leveraged sound effects, but with our large zoom jump from a single day to the Big Bang, we have often felt a zoom sound effect might be interesting to add.”

Further Reading

Bederson, B. The promise of zoomable user interfaces, *Proceedings of the 3rd International Symposium on Visual Information Communication*, Beijing, China, Sept. 28–29, 2010.

Cockburn, A., Savage, J., and Wallace, A. Tuning and testing scrolling interfaces that automatically zoom, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Portland, OR, April 2–7, 2005.

Igarashi, T. and Hinckley, K. Speed-dependent automatic zooming for browsing large documents, *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, San Diego, CA, Nov. 5–8, 2000.

Johnson-Stempson, R. ChronoZoom Tutorial, Microsoft Research, <http://research.microsoft.com/apps/video/default.aspx?id=161243>, March 12, 2012.

Ware, C. *Information Visualization, 3rd Edition*, Morgan Kaufmann Publishers, Burlington, MA, 2012.

Gary Anthes is a technology writer and editor based in Arlington, VA.

© 2012 ACM 0001-0782/12/12

In the Year of Disruptive Education

As college tuitions soar, various online models vie to educate college students worldwide—at no cost.

IN RETROSPECT, 2012 may well be remembered as the year when Internet technology enabled the popularity of MOOCs—or massive open online courses—a form of disruptive or transformative education currently growing at a meteoric rate.

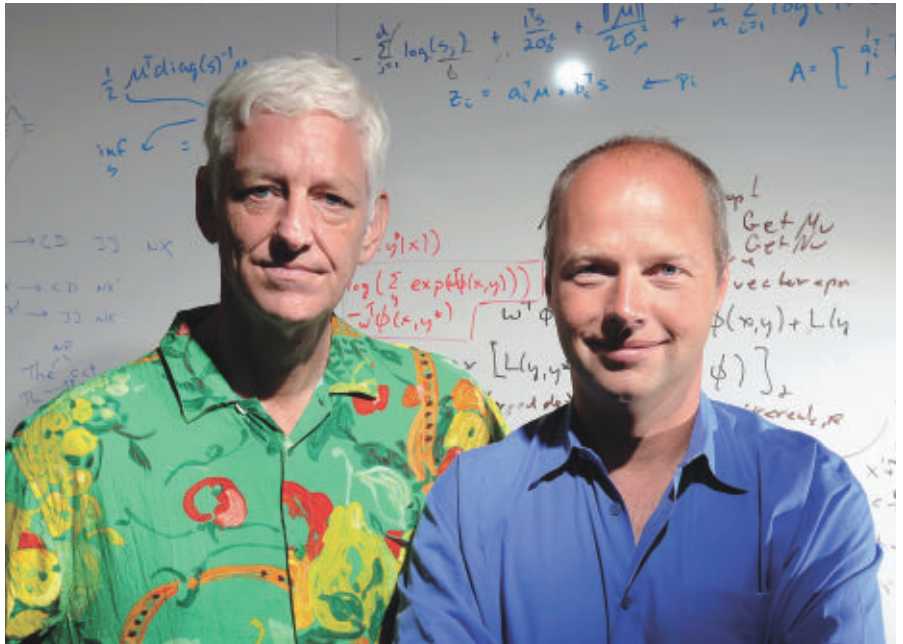
In just the past several months, top U.S. universities like Stanford, MIT, Harvard, Princeton, University of Michigan, and University of Pennsylvania put some of their classrooms online, giving countless students worldwide access to a higher education never before within their grasp.

One of the most successful online classes—some say it was the tipping point for the MOOC movement—was “Introduction to Artificial Intelligence” taught by Peter Norvig, Google’s director of research, and Sebastian Thrun, a Google vice president, last fall. More than 160,000 students signed up and 23,000 completed the course.

“There had been decades of various types of online classes,” says Norvig. “It is just that now all the technology is coming together to allow online classrooms of that size on a global basis.”

Those who may benefit most from MOOCs are the people in the developing world who previously had no access to consistent, quality secondary education at low cost. Indeed, of the students who took Norvig and Thrun’s class, one-third were from the U.S., one-third were from Europe, and one-third were from the rest of the world, with the majority located in India and Brazil.

Meanwhile, brand-new platforms such as Coursera, edX, and Udacity are committing themselves to making the best education in the world freely available to anyone who seeks it. There are no qualifications—other than an Internet connection.



Peter Norvig, left, and Sebastian Thrun’s “Introduction to Artificial Intelligence” class is often considered the tipping point for the popularity of massive open online courses.

But whether they achieve the level of success within the next five years that some educators are predicting depends entirely on how well the new technology-based method of teaching clears some very high hurdles.

“There is a crisis that has to do

How does one authenticate that the person doing the work is the person who signed up for the class?

with increasingly poor access to really high-quality learning opportunities,” says Richard Baraniuk, the Victor E. Cameron Professor of Electrical and Computer Engineering at Rice University. Thirteen years ago he launched Connexions, the initiative that offers free open-source textbooks via the Web.

“Everyone talks about the rising cost of health care,” says Baraniuk, “but it turns out that the cost of education is rising considerably faster. Student debt just crossed \$1 trillion in the U.S. and it is the number one form of debt now outside of mortgages. The problem is even worse in developing world. This latest open movement around courses, around textbooks, and around certificates is extremely exciting—and has a real chance of becoming a disruptive force in education.”

However, three issues have risen

to the top of a list of challenges for MOOC educators:

► **Business Models.** No one has yet determined how to make the MOOC model self-sustaining. Educators say the current emphasis is just on delivering content; the rest, they say, will work itself out “eventually.”

► **Student Evaluations.** How does an instructor effectively grade students who are doing homework and labs, writing essays, creating projects, and taking exams online? It’s easy when the tests are multiple choice or when the work is peer-graded, but technology still needs to catch up to evaluating the rest.

In the case of Norvig and Thrun’s AI class, the tests and quizzes were all multiple choice or fill-in-the-blank and could be graded automatically.

“It takes some effort to convert an open-ended question into a fill-in-the-blank question,” recalls Norvig, “but with some practice it can be done, at least for classes in technical subjects. For other, more open-ended subjects, like literature, one would have to use peer-assessment or self-assessment. Scott Klemmer of Stanford has a paper out [see the “Further Reading” list] in which he describes the successful use of self-assessment in the design course he taught.”

And, just as in physical classrooms, there is a trust issue: How does one authenticate that the person doing the work is the person who signed up for the class?

► **Certifications.** Because most MOOCs are not part of degree programs, what do students receive after completing a course? In most cases, it is a certificate. But is completing a free online course the same as having paid for and completed a course on campus? How potential employers will deal with this branding issue and how they will evaluate the meaning of, say, a certificate that testifies to a student’s taking an online Stanford course versus getting a degree from Stanford still needs to be played out.

“Students who put classes on their resumes who have not received official credit or degrees is, indeed, an issue,” says Norvig. “I think there will be a variety of accreditation schemes and we’ll just have to wait and see how valuable each of them turns out to be.”

“I think there’s a tremendous value to what we’re doing,” says Daphne Koller. “We just haven’t decided yet how to make it pay for itself.”

Meanwhile, Carnegie Mellon University, through its Open Learning Initiative, has been providing online learning for over a decade, with a focus on systematic student assessment.

“We’ve concentrated more on the science of learning than on delivering courses to the masses,” says Jeannette Wing, professor of computer science and department head of Carnegie Mellon’s computer science department. “But now, because the media has been focusing so heavily on course delivery, we have been having internal discussions about where we should be positioning Carnegie Mellon in this space. I can’t predict what decisions will be made, but I can say this is the number one topic on campus—just as it is the number one topic on many campuses right now, especially in the computer sciences.”

Full Speed Ahead

Other universities, however, are rapidly moving forward with delivering classes online, using various delivery platforms.

In mid-April, for example, two Stanford University computer science professors, Daphne Koller and Andrew Ng, launched Coursera, a for-profit “social entrepreneurship company” that partners with a growing number of universities—including Stanford, University of Michigan, Princeton, and University of Pennsylvania—and presents online many of the universities’ courses in the fields of computer science, biology, medicine, the humanities, and more.

“In October [2011], for the very first

time, Stanford made available online—and at no charge—three of its most popular computer science courses,” says Koller. “More than 300,000 students registered for ‘Introduction to Artificial Intelligence,’ ‘Introduction to Databases,’ and ‘Machine Learning.’ Seeing the success of that experiment, my colleague, Andrew Ng, and I decided we needed to build on it but couldn’t just make it a Stanford-specific effort. Great as Stanford is, there are other universities that offer amazing content currently to only a tiny sliver of the world’s population. So we decided we’d spin this out of Stanford and open the opportunities to other top institutions to engage with this amazing experiment on free access to top education for everyone.”

While Coursera was initially funded by a \$16 million venture-capital grant, Koller and Ng are still debating how to make their efforts sustainable.

“When we launched Coursera in mid-April, we had over 200,000 users,” says Koller. “Three months later, we had about 680,000—and it continues to grow. So I think there’s a tremendous value to what we’re doing. We just haven’t decided yet how to make it pay for itself.”

“Even if the content is to be free, which is something we care a lot about,” says Ng, “we believe it’s OK to charge a modest amount for certification—perhaps \$30 or \$50. Given the scale of numbers we have, we think we could come up with a sustainable business model if we decide to go that route.”

Another revenue-generating possibility, he says, is to charge prospective employers to facilitate introductions with Coursera students given the fact that, in IT, the cost per recruit in Silicon Valley can run as high as \$10,000.

Meanwhile, in May, MIT and Harvard each contributed \$30 million to edX, a non-profit joint venture by the two universities that, like Coursera, will offer free online courses to a worldwide audience. edX builds on MITx, a similar project launched by MIT this past December.

“My colleagues and I taught a prototype course on circuits and electronics this spring,” says Anant Agarwal, a professor of electrical engineering and computer science at MIT who is now president of edX. “Over 150,000

online students from 152 countries signed up for the course, and we gave out 7,200 certificates. Over 300 students scored a 100.”

edX is, in effect, an open source platform for universities worldwide that wish to offer courses online beginning this fall. MIT courses will be branded “MITx,” Harvard courses “HarvardX,” and other universities coming onboard will brand their courses “[university name]X.”

In addition to MIT and Harvard’s contributions, Agarwal is hoping to raise money from various foundations and philanthropies, while having similar thoughts to Coursera about the possibility of charging for placement services and/or for completion certificates.

“The question has arisen regarding the usefulness of such certificates,” says Agarwal. “We have talked to a number of potential employers and have anecdotal information that, if a student applies for a job and has an edX certificate with a good grade, employers will value the certificate and will count it in the assessment of the job candidate. I mean, MITx and HarvardX courses will be taught by the same professors who teach the on-campus classes, they won’t be watered down, and will be MIT- and Harvard-difficult to pass. Why shouldn’t the certificates carry value?”

Connexions, one of the very first open-education projects which began in 1999, is unlike Coursera and edX

edX is an open source platform for universities worldwide that wish to offer courses online.

in that its goal is not to offer courses but to build open source textbooks. Based at Rice University, the nonprofit platform enables authors to write textbooks that students can access free of charge. Funding comes from a number of philanthropies, including Rice and the William and Flora Hewlett Foundation.

“We currently have millions of users per month and our science, technology, engineering, and math content has been used about 100 million times in the last five years,” says Rice’s Baraniuk, who is both Connexions’ founder and director.

Baraniuk marvels at the number of disruptive education models that have sprung up in just the last few months, but won’t try to predict which will be the dominant model.

“All the models are extremely disruptive and are completely at odds

with today’s status quo,” he says. “Which is why they are all so exciting. There’s no question that the world of education is going to be fundamentally changed within the next 5–10 years. I’m convinced of that. I would also predict that the big incumbents are either going to have to transform their institutions or they’re going to be rendered irrelevant. The great opportunity here is that we’re going to be able to provide everyone—no matter where they are, whether they have money or no money—the ability to be able to learn and advance themselves in their lives and their careers.” **C**

Further Reading

Baraniuk, R.

Open education: One perfect storm yields three revolutions, *Telefonica*, May 12, 2012.

Bowen, W., Chingos, M., Lack, K., and Nygren, T. Interactive learning online at public universities: Evidence from randomized trials, *Ithaca S+R*, May 22, 2012.

Christensen, C., Aaron, S., and Clark, W. Disruption in education, *EduCause Review Online*, Jan. 1, 2003.

Coursera

Welcome to Coursera, <http://www.youtube.com/watch?v=PoJLL3E-zk0>, April 18, 2012.

Klemmer, S. and Kulkarni, C.

Learning Design Wisdom by Augmenting Physical Studio Critique With Online Self-Assessment, Stanford University technical report, July 2012.

Paul Hyman is a science and technology writer based in Great Neck, NY.

© 2012 ACM 0001-0782/12/12

Technology

PC Sales Decline, Laptops Evolve

Sales of personal computers dropped significantly in the third quarter of 2012, but analysts say the decline reflects an evolution in personal electronics, not the demise of the PC.

“I don’t think the PC is dying,” says Mikako Kitagawa, a principal analyst at Gartner, which reported worldwide PC sales down 8.3% to 87.5 million machines, compared to the same quarter last year. Analyst firm IDC estimated the quarterly drop at 8.6%, and iSuppli says total sales for the year will be

down 1.2%, the first annual decline since the dotcom bust of 2001.

Analysts attributed part of the drop to a slowing economy in China, the largest market for PCs, and continued turmoil in Europe. Manufacturers also held back sales to avoid tamping demand for machines featuring Windows 8, Microsoft’s new touch-screen optimized operating system. And although consumers do not buy tablets and smartphones to replace PCs, they are holding onto PCs longer and diverting dollars they

might once have spent for a new PC to a new iPad or data plan instead.

Craig Stice, senior principal analyst for the compute platform at iSuppli, says consumers with disposable income now ask, “Do I need a new PC or do I just want it? And if I don’t need a new PC, there’s a lot of things out there that might catch my attention.”

About half of the purchasers of PCs use them for business, producing spreadsheets, documents, PowerPoint presentations, tasks that are ill

suited for tablets. David Daoud, research director for personal computing at IDC, thinks people will use smartphones for communications, tablets for multimedia, and PCs for productivity. Meanwhile, PCs are changing their form. With the arrival of Windows 8, manufacturers are introducing hybrid laptops in which the screen can be detached and used as a tablet. “I think we’re reaching the stage where we’re seeing the PC evolve into a hybrid product,” Daoud says.

—Neil Savage

The Ultimate Online Resource for Computing Professionals & Students

ACM DL DIGITAL LIBRARY

<http://www.acm.org/dl>



Association for
Computing Machinery

Advancing Computing as a Science & Profession

Emerging Markets

IT Innovation for the Bottom of the Pyramid

New ways to develop technologies for the emerging growth markets.

THE AFFLUENT MARKETS of North America and Europe are tough places these days for IT. Competition is high, recession and uncertainty are never far away. It will be more logical for IT producers—big firms, small firms, universities, and so forth—to step outside of their comfort zones and focus on the mirror-image “bottom of the pyramid” (BoP). We can associate this generally with the emerging markets of low-income nations, or more specifically with that half of the planet’s population who live on less than US\$5 per day. These markets are high-growth—for example, mobile connectivity in Africa is growing by 30% per year and it is now the second-largest regional market after Asia²—and they have lower competition. Overall, BoP markets represent the next frontier for IT.

But along with new markets must come new ideas about how we innovate IT. Traditional models of IT innovation vary. The mythology is the lone inventor soldering bits of kit together in a garage, or cranking out code in an attic. The reality is more often a well-

funded R&D team based around an IT lab in California or Massachusetts.

Though itself now splintering, this model has historically worked well in producing the new hardware and software to sell to high-income consumers. But it struggles—even fails—to produce the right products for the low-income consumers of Latin America, Africa, and Asia. As examples, we can return, at best, an “open verdict” on transfer of the European-model “telecenter” and the U.S.-developed “One Laptop Per Child” to the developing world.^{1,4}

What is wrong with traditional models of IT innovation serving bottom of the pyramid markets? The simple answer is “design-reality gaps”: too great a difference between design expectations built into the IT and the developing world realities in which the IT is to be used.³ This typically arises because of the distance between innovators and users; not simply geographical distance but due to cultural, psychological, and linguistic differences.

In order to close those gaps, traditional approaches to IT innovation

will no longer suffice. This column outlines six emerging IT innovation models that are helping to close design-reality gaps, and which are successfully delivering new products for low-income markets. They do not represent fully distinct categories—some of them overlap. But they do represent ideas we need to adopt if IT is to fulfill its potential for the huge emerging markets of the world’s poor. Put more bluntly, if you do not adopt these models, someone else will. Indeed, someone else already is.

Intermediated Innovation

Large IT firms know their high-income markets well. But growth is happening elsewhere. How, then, to bridge that divide between what they know and what they need to know? One solution has been to engage with the new IT innovation intermediaries that have sprung up in recent years. They have a foot in both camps—a deep experience of how IT is used in low-income communities combined with an ability to understand the needs of big corporations.

Bangalore in southern India has become a hub for these innovation



A computer bazaar in Bangalore, India.

intermediaries. Some are employed directly by multinationals: Microsoft Research India (<http://research.microsoft.com/en-us/labs/india/>) has developed relations with a set of poor communities through which innovations can be piloted; learning from such pilot projects can then be fed back to other parts of Microsoft. Others are guns for hire: the Centre for Knowledge Societies (<http://cks.in/>) is a contractor that works on behalf of large private firms; using a mix of anthropological and technological study methods within poor communities, it can report back to its clients with key IT innovation pointers.

Working via these new intermediaries multinationals have been able to adapt their products, innovating to address the realities of bottom of the pyramid environments. In this way, mobile phones have emerged with greater battery capacity or antenna strength, to cope with the lower coverage of mobile and electricity networks experienced by rural users. But the established IT firms have taken a broad-brush approach, using a single cheap

model to serve markets across several continents. That has created an innovation gap that newly arriving Chinese firms have been quick to exploit.

Already better attuned to low-income customers than U.S. and European rivals, they have created informal innovation channels to intermediaries like wholesalers and distributors. This has enabled them to adopt much smaller niche innovation strategies. In Kenya, for example, Chinese firms dominate cellphone sales to the poor, with a 90% market share in some areas according to recent University of Man-

chester research. Working with their intermediaries they have introduced innovations including dual SIM card phones (allowing users to choose the lower-cost network to call particular contacts), translation of the phone interface into Swahili, and addition of a single-button-enabled new interface for the popular M-Pesa mobile money service.

Collaborative Innovation

In the intermediated innovation model, the intermediaries stand outside low-income communities, and the poor are respondents more than they are participants in the innovation process. A step further down the road toward inclusivity of innovation is a collaborative approach with greater involvement of those who will ultimately be using the new IT. This is mainly done through prototyping and a well-known example is the M-Pesa service, which began facing in one direction and ended facing in quite another. Local Vodafone subsidiary Safaricom worked with Kenyan partners in piloting a cellphone

Along with new markets must come new ideas about how we innovate IT.

application to help financial management in micro-finance institutions (organizations that manage micro-lending to the poor). The pilot did not work—the institutions disliked the new system and stuck with their existing manual methods.

But Safaricom designers noticed something unexpected: those provided with the service were converting cash into “m-cash” for security when traveling, and were then exchanging the m-cash to repay debts or provide financial assistance to family members. A eureka moment occurred, the project was reoriented, and M-Pesa was born as a service allowing users to transfer money from one cellphone to another. This was only possible because the firm’s IT innovators worked alongside developing country users, and learned from them.

Grassroots Innovation

The previous example exposes a broader point—put IT tools into the hands of the poor, and they will themselves start to innovate. This is “grassroots innovation”: not just innovation *for* the bottom of the pyramid but innovation *by* the bottom of the pyramid. Such innovations may be only minor but ethnographic study shows many IT-related appropriations and adaptations go on in poor communities:

- New processes. For example, beeping (or flashing) that allows a message to be communicated without the call being completed. Street vendors use this to receive free “I want to buy now” messages from known customers.

- New applications. For example, use of airtime as currency that allows cellphones to metamorphose into mobile wallets. Even before the advent of specialist mobile money services, those who own phones in poor communities have found ways to use them for payment (school fees, public transport, even childrens’ pocket money) or for receipt of remittances from distant relatives.

- New products. For example, backstreet recharging of phones. Informal-sector enterprises are emerging that strip and resell the circuitry from high-end phones, replacing it with basic calls-and-SMS-only functionality. They then sell the resulting high-end-body-with-low-end-innards as a unique hy-

brid for those who want the latest look but lack the budget to match.

Such innovations spread only haphazardly. They are much like memes—ideas that will mostly never see the light of day, but with a very few gaining momentum and being widely adopted. This is a huge pool of intellectual property: the IT equivalent of the great reserves of genetic biodiversity that lie within the tropical forests. And there will be similar moral questions about how these reserves are exploited.

Non-profit organizations like India’s Honey Bee Network or the multi-country Prolinnova program have been working for many years within poor communities to capture and share grassroots innovations; albeit with a focus to date on traditional rather than IT innovations. So inclusive models do exist. But large IT firms will increasingly recognize the value of grassroots innovations. They will then seek to formalize their currently somewhat jury-rigged connections with BoP markets, and will harvest, filter and diffuse such innovations on an industrial scale.

Frugal Innovation

An inherent characteristic of grassroots innovation—but also applicable to other models described here—is that it is “frugal.” Frugality can be seen as a characteristic of the output—an IT product or application that is not just low-cost but also low-demand in terms of other resource requirements, including electricity, telecommunications infrastructure, and human skills.

Multinationals can produce such items. The Nokia 1100—the world’s best-selling phone—was a classic example of closing the gap between IT design and resource-constrained BoP reality. Robustly built, it was pared down to the basic features that low-

income consumers could pay for, and included a much-valued component: a flashlight. Specific initiatives like Copenhagen-based “Frugal Digital” (<http://ciid.dk/frugaldigital/>) are also working in this space, developing a low-cost health screening meter designed like a standard alarm clock, a mobile device to test hearing loss, and a basic educational projector.

But output is only one aspect of frugality—it can also be a characteristic of the innovation process; when new IT is produced far from a high-tech, high-cost laboratory but in a low-price setting that is or imitates the context of its future use. Grassroots innovations are examples—frugal in both process and output—but difficult to capture. Of more immediate interest may be the frugal micro-innovations of developing country inventors who focus on local communities.

The World Bank’s infoDev project (<http://www.infodev.org>) has been a keen supporter of frugal micro-innovators, providing finance, incubator spaces, and training programs to help them bring their ideas to market. An example is M-Farm (<http://mfarm.co.ke>): a simple SMS-based app that allows farmers to buy and sell direct, cutting out the middlemen who often cheat them.

Innofusion

Though they may get to grips with designing IT for use by poor consumers, many of the examples in this column hold too narrow a view of what innovation is, equating it with invention, with the initial process of creating something new. But that is not enough to make an innovation effective—you have to move beyond the “build it and they will come” mentality and recognize the importance of diffusion and adoption of the new IT.

This is particularly so in the case of BoP markets. Distribution is not about negotiations with supermarket or retail chains: they do not reach low-income consumers in the global South. Those consumers operate in conditions of poor infrastructure and services, and are reached by individual, locally embedded micro-retailers and entrepreneurs.

But they too are innovators: the diffusion of a new IT product itself

An inherent characteristic of grassroots innovation is that it is “frugal.”

requires further innovations, so diffusion and innovation are bound together in a process we should call “innofusion.” What is needed are incremental alterations to the way the technology is configured, packaged, marketed, sold, used, supported, serviced, and so forth. And those alterations are context-specific; attuned to the specific needs and expectations of each community. This is not mass production, it is mass customization. And it is rarely innovation of the core technology; more often innovation of the social processes that are essential to IT diffusion and use.

Reverse Innovation

Some of these innofusion innovations make their way upstream in the value chain, and represent a small example of our final model: reverse innovation. Reverse innovation represents the mirror image of traditional concepts of innovation. Those traditional concepts see a flow of objects and ideas from the producer to the consumer; from the rich global North to the poor global South. Reverse innovation sees the opposite.

In 2012, development informatics researchers were amused to see British bank advertisements extolling the benefits of new services—such as PingIt—which would allow cellphone-to-cellphone transfer of money. In the U.K., we are used to measuring our technological lag to the U.S.: the time it takes for innovations to make it across the Atlantic. Now it seems the Mediterranean is the new crossing point: Kenyans, for example, have had mobile money transfer services since 2007. Thanks to the new innovation models, Africa is five years ahead of Europe, and the idea of m-money has flowed from poor countries to rich countries.

More prosaically, Cisco’s ASR901 routers were developed for the needs of service providers in emerging Asian markets, but then backflowed to U.S. and European clients. Their low-cost and energy-efficient designs arose from developing-country realities, but offered a competitive advantage also in more mature markets.

Conclusion

Innovation is a term so overused it can start to lose its meaning. But serving the world’s emerging markets—and perhaps lifting millions out of poverty

Serving the world's emerging markets will require us to “do new things in new ways”: the essence of innovation.

at the same time—will require us to “do new things in new ways”: the essence of innovation. More than that, it will require us to do innovation in new ways if we are to develop the new hardware and applications to achieve those goals.

For that to happen, we need a new mind-set. One that sees the world’s poor not in terms of the remote, helpless images often used by development charities, but through more positive and active images that understand them at least as IT consumers and perhaps, further, as agents in their own development.

For IT multinationals, universities, and innovators in North America and Europe, this change in mind-set may be particularly difficult given legacy attitudes toward the “Third World.” There are already signs that IT firms in China and India are more adept at working with innovation intermediaries and at frugal innovation. Absent more change in the global North, it may be these mid-income nations that capture the next technological frontier that the bottom of the pyramid represents. C

References

1. Best, M. and Kumar, R. Sustainability failures of rural telecentres. *Information Technologies and International Development* 4 (2008), 31–45.
2. GSMA. African Mobile Observatory. GSMA, London, 2011.
3. Heeks, R. Information systems and developing countries: Failure, success and improvisation. *The Information Society* 18 (2002), 101–112.
4. Severin, E. et al. Evaluation of the “Una Laptop Por Niño” Program in Peru. IADB, Washington, D.C., 2011.

Richard Heeks (richard.heeks@manchester.ac.uk) is the director of the Centre for Development Informatics at the University of Manchester, U.K.; <http://www.cdi.manchester.ac.uk>.

Copyright held by author.



ACM's
interactions
magazine explores
critical relationships
between experiences, people,
and technology, showcasing
emerging innovations and industry
leaders from around the world
across important applications of
design thinking and the broadening
field of interaction design.
Our readers represent a growing
community of practice that
is of increasing and vital
global importance.

interactions
<http://interactions.acm.org>



Historical Reflections

Saving Private Gromit

Reflections on the legalities and economics of preserving animations and games in Europe.

THIS PAST MARCH, I attended the “Westminster Media Forum Keynote Seminar: The state of play—next steps for the U.K. video games industry.” The previous day the U.K. government had announced a series of tax breaks for the animations and games industry, and there was a real sense in the room that video games were finally taking their place in the cultural heritage landscape.

The emergence of games as cultural objects poses some interesting questions both for the industry and for government. There is an indisputable public interest argument for preserving for future generations those objects that we take to have cultural significance. Games, often dismissed as mere “entertainment,” nevertheless speak to a very wide international audience and express something profound about the current zeitgeist. To that extent, at the very least, they are preservation-worthy. But what will preserving them involve, and who is going to pay the preservation piper?

An obvious suggestion is that games ought to be preserved by the libraries, archives, and museums that take responsibility for preserving, curating, interpreting, and making available to the general public the rest of our cultural heritage. The difficulty is that preservation is never for free and in the case of complex digital objects it can be very expensive. Indeed, the cost of migrating a game from one hardware platform to another—an activity that might be necessary many

times over the preservation lifespan of a digital object—might easily cost as much in time and money as developing the game from scratch. This assumes, of course, that memory organizations have at their disposal people with the necessary skills to complete the task; this activity would need to be carried out on an industrial scale. These are difficult times, but even if they were not, there is every reason to believe finance ministers might find higher priorities on which to spend tax dollars than preserving Wallace & Gromit, Grand Theft Auto, Tomb Raider, and Pong.

Another significant limiting factor in memory organizations taking responsibility for preserving digital material is the series of obstacles resulting from the complicated and

often contradictory legislative landscape under which preservation activity takes place. Of most immediate concern to preservationists is the national legislation under which they operate day to day. Different nation-states have their own laws and the understanding of key terms that prevails in one country often does not conform to that which holds elsewhere. In the U.K. context, attention needs to be given to the European Community framework law, which supersedes national legislation, and which, although meant to be incorporated into member state legislation, is neither uniformly or completely implemented across the whole of the EU. Here too, there is some disagreement over the interpretation of key terms. Finally, there is non-EU legislation, and international treaties and obligations such as the Paris Convention for the Protection of Industrial Property (1883), and the Berne Convention for the Protection of Literary and Artistic Works (1886), to consider.

Key legislation at the Community level includes:

- ▶ Directive 2001/29/EC of May 22, 2001 on the harmonization of certain aspects of copyright and related rights in the information society (the “Information Society Directive”).
- ▶ Directive 2009/24/EC of April 23, 2009 on the legal protection of computer programs (Codified version replacing the abrogated Directive 91/250/EEC of May 14, 1991, known as the “Computer Programs Directive”).
- ▶ Directive 96/9/EC of March 11,



IMAGE COURTESY OF DAVID ANDERSON

1996 on the legal protection of databases (the "Database Directive").

(Collectively referred to as the "Community Framework.")

► Directive 2001/84/EC of the European Parliament and of the Council of September 27, 2001 on the resale right for the benefit of the author of an original work of art.

► Directive 92/100/EEC of November 19, 1992 on rental right and lending.

Problematically for the anyone intending to preserve games, the Community Framework does not recognize the notion of multimedia works as a specific type of protected content, and no definition or specific framework related to multimedia works is available in EC law. Reproduction of multimedia works is addressed at the Community level through the various copyright and related rights directives as they apply to the constituent elements of a multimedia work: for example, software programs, databases, sound, and images. This is a pattern that is replicated in national legislation and in practice this means a distributive, fragmented approach is adopted, within which each component part of a multimedia work must be considered separately. Since multimedia works are not, in general, made available on computer platforms in such a way that individual elements can be removed from the whole, this means a multimedia work will effectively enjoy, as a whole, the strongest protection under law that is available for any of its constituent parts.

Taking a legally distributive (or cumulative) approach to multimedia works means each individual multimedia work a library or archive intends to preserve must be considered in order to determine the protection that it enjoys under the law, and what exemptions might apply. It is not legally safe to assume every game enjoys the same protection. As the law protects each element of the multimedia work separately, games with music will have a different legal character than those without, and so on. This makes it effectively impossible to carry out legal due diligence on the scale required.

European law grants libraries and archives no general right of reproduction but permits them to reproduce (or transfer) digital material only in

In view of the financial and legal difficulties inherent in public bodies attempting to preserve games, might it be possible to place this responsibility on the industry?

certain specified cases. While the exemptions libraries enjoy are sufficient to permit some of the activities necessary for preservation, they fall far short of covering the full range required. There is a tendency for national legislation to be both more permissive than Community law, and for it to provide a greater degree of detailed governance. This flatters to deceive however, as national regulation is often simply incompatible with Community law, and is open to be overturned by anyone minded to press the point at the European level.

In view of the financial and legal difficulties inherent in public bodies attempting to preserve games, might it be possible to place this responsibility on the industry? This would have the immediate advantage of circumventing many of the difficulties outlined in this column. After all, copyright holders have the absolute legal right to preserve the material they own, and arguably have at their disposal better human and financial resources to undertake this task than do public bodies.

Games represent an important sector of the modern economy. In the U.K. alone, there are around 200 games studios, employing around 28,000 people and generating global sales of around \$2.65 billion a year. The industry contributes approximately \$623 million per annum to national GDP. How-

Calendar of Events

December 17-18

The Eighth Asia Information Retrieval Societies Conference, Tianjin, China,
Contact: Yuexian Hou,
Email: krete1941@gmail.com

December 18-20

Third Kuwait Conference on e-Services and e-Systems, Kuwait,
Contact: Saleh Kassern,
Email: kasserns@yahoo.com

January 3-6

International Conference on Distributed Computing and Networking, Mumbai, India,
Contact: Shyamasundar Rudrapatna,
Email: shyam@tifi.res.in

January 6-8

ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA,
Contact: David S. Johnson,
Phone: 908-582-4742,
Email: dsj@research.att.com

January 7-10

Fifth International Conference on Communication Systems and Networks, Bangalore, India,
Contact: David Kotz,
Phone: 603-646-1439,
Email: kotz@cs.dartmouth.edu

January 9-12

Innovations in Theoretical Computer Science, Berkeley, CA,
Contact: Satish Rao,
Phone: 510-642-4328,
Email: satishr@cs.berkeley.edu

January 11-12

Annual Symposium on Computing for Development, Bangalore, India,
Contact: William Thies,
Email: thies@microsoft.com

January 16-20

Foundations of Genetic Algorithms XII, Adelaide, Australia,
Sponsored: SIGEVO,
Contact: Frank Neumann,
Email: frank.neumann@adelaide.edu.au

ACM Journal on Computing and Cultural Heritage



JOCCH publishes papers of significant and lasting value in all areas relating to the use of ICT in support of Cultural Heritage, seeking to combine the best of computing science with real attention to any aspect of the cultural heritage sector.

www.acm.org/jocch
www.acm.org/subscribe



Association for
Computing Machinery

ever, this does not mean the games industry can be expected to bear the cost of games preservation. The success of the industry in securing fiscal concessions at a time of widespread austerity is evidence of the competitive imperatives in this industry, and bears testament to the effectiveness of their lobbyists. The barriers to entry in the games development domain are now so low, and the infrastructural requirements so minimal, that it is relatively simple for studios to set up for business anywhere in the world, or to relocate from one country to another in a way that is impossible for traditional “smokestack” industries. The industry is in the middle of a transition from selling relatively expensive games on shiny discs in shrink-wrapped packaging to providing inexpensively produced (and priced) content for download from the cloud. Not only is this shortening supply chains and undermining the business models of traditional companies, but it is leading to the emergence of new games studios that are smaller, leaner, and more dynamic than anything that has gone before. It also means that any government intending to pass onto the games industry the costs associated with preservation will not be in a strong position to impose its will.

There are a number of steps that can be taken on an international level that would do much to improve the prospects of ensuring games preservation in the public interest takes place without imposing undue burdens on government or the industry. The final report of the Blue Ribbon Task Force on Sustainable Digital Preservation sets out four key recommendations:

- ▶ Leading cultural organizations should convene expert communities to address the selection and preservation needs of commercially owned cultural content and digital orphans.
- ▶ Regulatory authorities should bring current requirements for mandatory copyright deposit into harmony with the demands of digital preservation and access.
- ▶ Regulatory authorities should provide financial and other incentives to preserve privately held cultural content in the public interest.
- ▶ Leading stewardship organiza-

It is not legally safe to assume every game enjoys the same protection. This makes it effectively impossible to carry out legal due diligence on the scale required.

tions should model and test mechanisms to ensure flexible long-term public-private partnerships that foster cooperative preservation of privately held materials in the public interest.

To this final recommendation, I would add the suggestion that the industry should be encouraged to make available as much information as possible on the hardware and software dependencies of their products. This is neither commercially sensitive nor difficult to provide at the time when software is released. It is, however, absolutely vital in ensuring long-term access to preserved digital material, and becomes increasingly difficult to uncover over time. It would be possible to seamlessly incorporate harvesting this information along with the details the industry already provides to age rating agencies such as PEGI.

Announcing the games and animation industry tax breaks, U.K. Finance minister George Osborne told Parliament it was his intention to keep Wallace & Gromit exactly where they are. With the proper level of cooperation between the industry and government it should be possible to save Wallace & Gromit, Laura Croft (and the rest) for future generations. C

David Anderson (cdpa@btinternet.com) is the CITECH Research Centre Director at the School of Creative Technologies, University of Portsmouth, U.K.

Copyright held by author.



Kode Vicious

Can More Code Mean Fewer Bugs?

The bytes you save today may bite you tomorrow.

Dear KV,

One of the coders I work with keeps removing my calls to `system()` from my code, insisting it is better to write code that does the work I am doing via the shell. He keeps saying it is far safer to code using the language we are using than to call out to the shell to get this work done. I would believe that if he did not add 10 to 20 lines of code just to do what I do in one line with `system()`. How can increasing the number of lines of code decrease the number of bugs?

Happy with the One-Liner

Dear One,

You almost had me with your appeal to simplicity, that having a single line with `system()` on it reduces the potential for bugs. Almost, but not quite.

When you call out to the shell from any language, you are not using a single line of code, but thousands. Calling a shell at this point is like using a nuke to kill a flea. That flea will be very dead when you are done, but you have also wasted a lot of energy in killing it, and it may result in collateral damage. Each and every call to `system()` is trusting all that underlying code, and the issue is not only that there are a lot of lines under there, but also that the things the shell can do are extremely powerful—it is probably the most powerful program on any system.



A command shell on any system—Unix, Windows, or otherwise—is there to command the system, and it has accreted to itself, over time, the ability to do things to the system in a single line that really should require a bit more thought. The obvious example is moving and removing files. I would actually like to think that most programmers know better than to do something like calling `system()` with an `rm` command, in particular one in which they supply unchecked user input to the call to `system()`. While I say “I would like to think that,” a voice—a very loud voice—in the back of my head is screaming, “It’s not so! It’s not so! They’ll do it!

Stop them now!” I hate that voice, but no matter how I try to drown it out—and I have tried—there it remains.

A worse offense than invoking the `rm` command in a system command is calling a shell script via `system()`. Why? Because the poor sap reading your code later will probably have no idea what the script does. I am sure it will have a descriptive name such as `update_sales_2.sh`, and that it will be checked in to the source repository rather than being stored in the `/home/bob/update_test2/` directory, but perhaps it won’t, and the fragile setup I describe here will ring true: when Alice comes to read the code, she will then have to go read `update_sales_2.sh`, so long as she has read access to Bob’s home directory. She will be reading it at 3 A.M. when something has broken, and this will all go well and everyone will live happily ever after.

Perhaps my very favorite abuse of the `system()` routine in code is when it is used to build up a complex pipeline of commands. Using pipes in a call to `system()` in a program is a fine way to launch a fork bomb into your system, especially if the pipes are built up on the fly.

Running a complex piped command from a shell by hand has less risk of hammering the machine because humans are slow—and supposedly they are paying attention to what they are doing. Once you put a set of

pipes into `system()` and then let your code run unattended, you run the risk, should there be a bug in your pipeline, of repeatedly forking subprocesses and overwhelming whatever machine you are running on.

In the shell's defense, it does handle pipelines better than most programming languages, as anyone who has tried to use pipes and signals in C would readily acknowledge, but forking processes automatically must be done with great care. Too many times I have seen coders work out a nice pipeline incrementally in their terminal windows, and then cut and paste that into a program that will be executed not by hand, but by another program. The look on their faces when the system running their pipeline in a loop destroys their system is somewhat amusing, but hardly makes up for the fact that they, or their whole work group, are about to lose work because a system reset is necessary.

Finally, and this is probably the most important and most subtle argument against using `system()`, doing so requires the programmer reading the code to mentally context-switch from one language to another. Unless you are calling `system()` in a shell script, the language that the programmer is reading when reaching the call to `system()` is very much not shell; it is C, C++, Python, Perl, Ruby, or something else. That means all of the mental context you have built up while working on the code is about to be lost as you bring in the shell-scripting context, or you are simply going to gloss over it and make a mistake because you are not thinking in shell when you get to the call to `system()`.

It is not that this cannot be done, but it definitely increases the cognitive load on the reader, so you had better have a very good reason for switching into the shell—something better than not wanting to figure out how the unlink system call works.

KV

Dear KV,

Why do some modern network protocols not have sequence numbers? I would think that by now all protocol designers would have realized that having a simple sequence number in each

packet helps people in debugging their network setups.

Out of Sequence

Dear OoS,

You might as well ask why people insist on not wearing seat belts after all of the years that particular technology has been proven to save lives. People will, it seems, persist in the optimistic belief that everything will be OK so long as they are otherwise careful. They think that bad things happen only to other people's protocols, or packets, but not to theirs.

I want to make two points in response to your plea for sanity in network-protocol design. The first is that it is not just having a sequence number that is important, but how the sequence number is used is important as well. Consider the sequence number in TCP, which counts the bytes that have been communicated between two endpoints. When TCP was designed, the fastest network in common use was a 10Mbps Ethernet LAN. Please note that is an M, not a G—10 megabits per second. At 10Mbps, transmitting 2^{32} bytes of data takes approximately 3,400 seconds, or just less than an hour, which is an eternity to a computer. On commodity 10Gbps hardware available today, it takes 3.4 seconds to transmit the same data, meaning the sequence space rolls over about every four seconds. If a packet is lost for more than four seconds, there is a nonzero probability that data on the connection will get munged. With hardware that will be available quite soon, the time will drop to 0.3 seconds for the sequence space to roll over.

None of this is to say that TCP was poorly designed (heck, at least it had a sequence number), but it is important for designers of modern protocols to understand the future proofing-vs.-space trade-offs when selecting a sequence number. If at some point TCP is extended, then the sequence number could be increased to 64 bits, which even at 100Gbps would require 46 years to roll the number over. Any packet lost in the network that long will be quite lost indeed. When you choose a sequence number, consider what you are protecting. With TCP it is protecting all the bytes transmitted, so that none is lost or reordered on delivery. With

other protocols it might be necessary to count whole messages only, so that the receiver can say that packet A arrived before packet B rather than worrying about every byte in the message.

The second point I would like to make is that timestamps are not good sequence numbers. While it is common to believe time always moves forward, this is often not the case in computing. Many bugs crop up in dealing with time on computers, not the least of which is that different clocks, on different computers, often proceed at different paces. This is why we have protocols such as NTP (Network Time Protocol) and PTP (Precision Time Protocol) to discipline our computer clocks. Alas, computers do not like to be disciplined, and even when running a time protocol, the clocks on two computers are always somewhat offset from each other, so running a time protocol does not solve this problem. Leaving aside the mind-bending relativity problems of computer timekeeping—and trust me, you really want to leave those aside—the fact remains that using the time on a computer as a packet sequence number is problematic. Incrementing a counter is easier, faster, and less error prone than making sure that the timestamp you received is monotonically increasing. For the case of packet sequencing, simpler is better—and simpler is a counter.

To those who design or hope to design network protocols, please, I beg of you, do not skimp on the sequencing numbers. The bytes you save today will bite you tomorrow.

KV

**Q Related articles
on queue.acm.org**

Kode Vicious to the Rescue

George Neville-Neil

<http://queue.acm.org/detail.cfm?id=1035604>

Languages, Levels, Libraries, and Longevity

John R. Mashey

<http://queue.acm.org/detail.cfm?id=1039532>

Self-Healing in Modern Operating Systems

Michael W. Shapiro

<http://queue.acm.org/detail.cfm?id=1039537>

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

Copyright held by author.



The Profession of IT Moods

Recognizing and working with moods—your own, your team's, and your customers'—is essential to professional success.

COOPERATION, IT IS said, is a stronger force in nature than competition. To get things done, we need to be good at collaborating and coordinating. Competition is necessary with rivals, but competition with customers, clients, and teammates will not get us very far.

You have probably heard dozens of tips about the mechanics of collaboration and coordination. One thing you have probably not heard much about is moods. Mastery of moods—your own, your customer's, and your team's—is essential to your success.

Imagine yourself as a project manager. One of your ideals is clarity: your requests clearly state the desired outcome and due date. But is clarity enough? If you make your request to Jane, who is in a cheerful mood, she will accept your request gladly. But if you ask Mike, who in his mood of resignation sees no possibility that your desired outcome is achievable, you may find yourself in an argument about the futility of the request. Or if you ask Leslie, who is in a mood of resentment about perceived injustices by teammates, you may encounter a begrudging acceptance followed by covert attempts to undermine your project. In short, the same request leads to different negotiations and different actions depending on the recipient's mood.

Or imagine that your team just suffered a setback because customers did not react enthusiastically to the new software. You ask team members for their evaluation of what has happened. Caitlin says, "I don't know what is going



ing on. But I would love to interview our customers and find out what was behind their reactions. I am certain I will learn something that will help improve our software." Harry says, "I also don't know what the heck is going on. But I do know those customers are jerks. We listened to their requirements and delivered what they asked for. And this is how they treat us?" Which of those two is going to be helpful to finding the mistake and fixing it? Both say they do

not know what is going on. But each embeds that acknowledgment in another mood that shapes their attitude toward future actions. Caitlin brings curiosity, a mood of openness to learning from the situation. In contrast, Harry brings confusion, a mood of hostility toward not knowing what is going on. Team leaders wish that everyone brought the mood of curiosity, or even wonder, when it is time to learn something new. Teachers wish the same for

their students. It is so much easier to learn from your mistakes and move on in a mood of curiosity. Can such a mood be cultivated?

Moods belong not just to individuals, but to groups. Your mood is often shared with others around you. Speakers try to read the “mood in the room”—a sense that the audience as a whole has an overall receptivity or hostility to what the speaker has to say. Managers pause when their teams or organizations have poor morale; they know they will have trouble accomplishing their mission unless they can instill team spirit. Marketers pitch advertisements to their sense of the public mood.

These examples remind us that we are constantly affected by our own moods and the moods around us. But we are not always aware of moods and we do not always know how to respond to and manage them. My purpose in this column is to help you recognize moods and interact effectively with them.

Moods Defined

Moods and emotions are related but are not the same. Emotions are feelings that individuals experience in response to various stimuli, in the

context of their individual histories. Psychologists recognize eight basic emotions, with each positive balanced by a negative, as follows:

- Love – Hate
- Joy – Sadness
- Peace – Anger
- Curiosity – Fear

Each emotion has multiple shades corresponding to different intensities of the feeling and the disposition toward acting on the feeling. For example, a thesaurus shows at least 40 synonyms for anger, representing different degrees of anger from annoyance and irritation on the low end, to chagrin in the middle, and resentment, rage, and revenge on the high end.

You can find many quick tabulations and maps of emotions on the Web.^a Robert Roberts⁴ and Robert Solomon⁵ both give deeper, insightful accounts of emotions.

Moods are general pervasive states of interpretation about the world. They act as “filters” for seeing the world. They incline toward certain emotions and actions and away from others.

a Numerous Web resources tabulate and categorize emotions and moods, for example, <http://www.scribd.com/doc/3850260/Map-of-the-Emotions>.

Some emotions sustained over a period of time can be moods; for example, joyful is a mood in which one experiences constant joy. The main features of moods are:

- Social. They belong to groups as well as individuals. In contrast, emotions are individual responses.
- Physical. They are accompanied by feelings and sensations. For example, tightness of chest when in anxiety, fatigue when in resignation, and muscle tension when in resolution.
- Outlook. Interpretation is positive (energizing), neutral, or negative (de-energizing). The interpretation tells us what is possible or not possible.
- No identifiable trigger. You find yourself in a mood but you cannot identify an event that put you there or even experience the transition into the mood. It just happens. Yet, moods are malleable: certain conversations, as well as music and poetry, can bring on a mood.
- Linguistic indicator. Your internal and external conversations give strong clues to moods.^{2,3}

► Cultivation. You can cultivate moods so that you spend more time in good ones and less in bad ones. For example, a daily practice of expressing gratitude can cultivate a mood of gratitude.

Table 1 lists examples of moods commonly encountered in the workplace. The table includes sample linguistic indicators for each mood.

Being Wise in Interactions

How does my mood interact with people I am managing? Leading? Teaming with? The answer depends on the moods and your skill at interacting with them. If you and the other person are in positive moods, the interaction is likely to go smoothly and you can achieve the desired outcome. If you both are in negative moods, a stand-off or some sort of losing outcome is likely. If one is positive and the other negative, the outcome is much less certain. Which person’s mood will most affect the outcome? A person skilled in working with moods may be able to identify and then shift the other’s mood to positive. For example, a good manager will:

- Help the resigned person by initiating a conversation for possibilities.

Table 1. Common workplace moods.		
Type	Mood	Linguistic Indicator
Positive (energizing)	joy, gratitude	Sees many things as gifts and is delighted and grateful for them.
	speculation	Sees new possibilities and likes them.
	ambition	Strongly desires an outcome and is committed to achieving it.
	resolution	Determined to make the outcome happen, no matter what.
	wonder	Does not know what is going on, and enthusiastically welcomes the opportunity to learn more.
Neutral	curiosity	Has a strong, perhaps overwhelming, desire to find out what is going on.
	inquiry	Does not know what is going on, and will keep looking until an answer is found.
	perplexity, bafflement	Unable to make sense of what is going on.
Negative (de-energizing)	apathy	Indifference toward the proposal and no motivation to try.
	confusion	Does not know what is going on, and annoyed or hostile about the lack of knowledge.
	overwhelm	Frustrated by the sheer volume of what is going on, and cannot figure out what to do.
	resignation	There are no possibilities for resolving the issue.
	resentment	The person perceives insults or injuries and quietly and resolutely seeks retribution; refuses to discuss.

► Help the overwhelmed person lessen the perceived overload by sorting through commitments and dropping the least important.

► Help everyone avoid the toxic moods of distrust and resentment by coaching them to avoid commitments they cannot fulfill, keep the ones they make, and take care of others' interests.

► Help end confusion by bringing others into moods of curiosity or wonder.

The bottom line is this: By entering into conversations associated with the mood you would like, you can take the other person there with you.

Collaboration and Competition

Collaboration is a strong human force. In collaboration, you and the others share a desire for an outcome and are committed to working together to achieve it; no one of you can do it alone. For collaboration to work, the group members need moods of appreciation, trust, and mutual commitment to the mission. Teams and networks good at collaboration are most likely to achieve their goals.

In competition, you and your rivals vie for an outcome that only one of you can have. You get it by defeating or outperforming the others in some way. Your team may distrust your competitors. To win a competition, your team needs moods of focus and discipline.

Collaboration and competition are complementary moods with strong interplay. In professional life, for example, your projects may be competitions with other teams or companies. But your team, company, or profession will continue on when individual projects are done. Your skill at recognizing and reconciling the moods for collaboration and competition is a definite professional asset. Hone it well.

Moods for Teams

Richard Strozzi-Heckler⁶ describes four stages of project evolution on a team as follows:

Formulation. You invite a group of people to help you accomplish a mission. You help them develop mutual appreciation and personal commitments to work together for the mission.

Ramp-up. You establish team discipline to meet commitments and start producing results; a lot of discipline and determination are needed because

Table 2. Moods and team project cycles.

Stage	Supporting Moods	Detracting Moods
Formulation	speculation, wonder, ambition	resignation, confusion
Ramp-up	discipline, focus, destiny, perseverance	apathy, indifference, lack of focus
Containment	appreciation, trust, resolution, "zone"	resentment, distrust
Completion	gratitude, forgiveness of debts	lack of appreciation, grudges

few results will be apparent yet and mistakes will be costly. You constantly remind of the value produced by the mission. You help team members learn to work together and trust each others' actions. To improve team performance, you help them share assessments about individual performance. You help establish needed outside relationships (with suppliers, for example).

Containment. You help the team stay focused on its mission, execute its commitments, cope well with the occasional breakdown, produce good results, and satisfy its customers. In some cases your team will enter the "zone," a higher state of perception where the results seem to flow without much effort and every coordination goes seamlessly.

Completion. You help the team declare that its work is done, wind down its remaining commitments, and celebrate the end of the project.

The effective leader inspires the team to the moods conducive to each stage; otherwise the team will not make it to the next stage (see Table 2).

The progression through these stages is difficult to achieve without solidarity—a mood of mutual trust, taking care of each other, and willingness to stand firm in unity against adversaries. Personal moods can coexist with the group mood of solidarity. Solidarity is a coveted mood. Leaders can cultivate it by instilling five basic practices into the team¹:

► Respect everyone's differences.
 ► Effectively make and coordinate commitments that produce value for others.
 ► Listen for opportunities to bring value to others.

► Build trust with others by making assessments that facilitate taking care of each other's concerns.

► Observe and bring to the foreground underlying moods that may help or hinder the ability to act with and listen to others.

The last two practices explicitly rest on skills with emotions and moods. A leader's skills can be put to the test when some team members fall into distrust, resentment, or other negative moods. If the leader cannot resolve the negative moods, the team may fail.

Conclusion

The moods and emotions of the people around us—our partners, teams, and groups—strongly affect performances. Positive moods enhance performance; negative moods detract and can render teams and groups dysfunctional.

Skilled managers, facilitators, and teachers are keenly aware of moods and emotions. They know how to interact effectively with different moods and can guide their groups through the moods necessary for a successful project, problem resolution, or learning.

The conclusion for us as professionals is that we have to continue to develop our own sensitivities to moods and emotions. We rely on our individual skills to manage moods in our teams and groups. 

References

1. Denning, P.J., Flores, F., and Flores, G. Pluralistic coordination. In *Business, Technological, and Social Dimensions of Computer Games: Multidisciplinary Developments*. M.M. Cruz-Cunha, V.H. Varvalho, and P. Tavares, Eds., 2011, 416–431; doi: 10.4018/978-1-60960-567-4.ch025.
2. Denning, P. and Dunham, R. *The Innovator's Way*. MIT Press, 2010.
3. Flores, F. The leaders of the future. In *Beyond Calculation*. P. Denning and R. Metcalfe, Eds. Copernicus, 1997, 176–192.
4. Roberts, R. *Emotions*. Cambridge University Press, 2003.
5. Solomon, R. *True to Our Feelings*. Oxford University Press, 2007.
6. Strozzi-Heckler, R. *Anatomy of Change: A Way to Move Through Life's Transitions*. North Atlantic, 1984, 1993.

Peter J. Denning (pjd@nps.edu) is Distinguished Professor of Computer Science and Director of the Cebrowski Institute for information innovation at the Naval Postgraduate School in Monterey, CA, is Editor of *ACM Ubiquity*, and is a past president of ACM.

I am grateful to Fernando Flores who, in 1987, introduced me to linguistic signatures of moods and emotions, and inspired me to learn more.

Copyright held by author.

Broadening Participation

Why We Need an ACM Special Interest Group for Broadening Participation

A proposal for an international group focused on broadening participation.

OPENING REMARKS TO the *Communications Broadening Participation* column series set forth the reasons why broadening participation is important to the computing field—the increasing worldwide demand for computing professionals, the importance of equal opportunity, and the significance of diversity for driving creativity.³ Since then, computing and informatics continue to drive innovation and economic growth in almost every societal and business sector, exacerbating the need for a larger, more diverse computing workforce. Yet there continue to be disparities in the participation of women, persons with disabilities, and underrepresented minorities in computing, in the U.S. and globally. Although the declining enrollments experienced by U.S. computer science undergraduate programs since 2000 have finally reversed, the participation gap persists (see National Science Foundation data, <http://www.nsf.gov/statistics/wmpd/>). In this column, I argue for the creation of an ACM Special Interest Group for Broadening Participation (SIGBP) to help solidify the BP community and amplify advances made by other BP organizations that only focus on a piece of the problem.

BP Community

A community for broadening participation has been built up over the past de-



cade thanks to the efforts and support from the ACM, IEEE, the Computing Research Association, and the National Science Foundation, among others. The community is interested in research and innovation to identify new approaches for BP as well as in implementation to replicate and scale up effective practices. The community is raising the awareness of the need for BP and providing professional development to support students and professionals.

For example, the ACM's Committee on Women in Computing (ACM-W) sponsors events, supports ACM-W student chapters, and collects data to

advance women in computing. The Computing Research Association's Committee on the Status of Women (CRA-W) similarly supports women's participation in computing research. ACM, CRA, and IEEE co-sponsor the Coalition to Diversify Computing (CDC)—which organizes the Tapia Celebration of Diversity in Computing conference for college students. CDC also sponsors Collaborative and Distributed Research Experiences for Undergraduates programs (CREU and DREU) to encourage underrepresented undergraduates to continue on to graduate school (see <http://www.cdc-computing.org>).

In 2001, NSF encouraged computing researchers to address the need for BP by introducing the Information Technology Workforce (ITWF) program and housing it within the Computer & Information Science & Engineering (CISE) Directorate. ITWF called for systematic research efforts to address the underrepresentation of women and minorities in computer science and engineering. In 2005, NSF CISE established the Broadening Participation in Computing (BPC) program, calling for large alliances to design and carry out comprehensive programs for BP. The BPC program funded 13 statewide or national alliances and over 100 smaller projects. The important impact being made by the alliances was summarized in this column last year.¹ From 2006–2010, NSF hosted annual BPC Community meetings with over 300 participants from various sectors.

Two national organizations advance the participation of women in computing. The Anita Borg Institute for Women and Technology (ABI) hosts the Grace Hopper Celebration of Women in Computing (GHC), an annual conference with over 3,000 student, faculty, researcher, and industry attendees (see <http://gracehopper.org>). The National Center for Women and Information Technology (NCWIT) builds networks through its Academic Alliance, Workforce Alliance, and K–12 Alliance to broaden the participation of women (see <http://www.ncwit.org>). The Center for Minorities and People with Disabilities in Information Technology (CMD-IT) is emerging to provide a similar community for minorities and persons with disabilities (see <http://www.cmd-it.org>).

The ACM, IEEE Computer Society, NCWIT, Computer Science Teachers Association, Microsoft, and Google are guiding an advocacy coalition called “Computing in the Core” to raise awareness among federal and state policymakers about the need for a larger more diverse computing workforce. The group promotes a set of principles underpinning K–12 CS education, including that “ethnic and gender diversity in the information technology field and computer science classrooms is crucial to the long-term success of the field” (see <http://www.computinginthecore.org>).

What is needed is an overarching community to connect, support, and amplify BP initiatives worldwide.

Proposed SIGBP Rationale

Given the infusion of resources invested in BP over the past decade, one would expect the participation gap to be shrinking. While the literature reports positive outcomes from individual BP initiatives, collectively we are not closing the gap on a national or international level. I suggest the key reasons include lack of coordination to create a strategy for widespread change, lack of a referencing mechanism to enable BP efforts to build upon prior work, and lack of awareness or appreciation of BP issues.

Coordination. There is strong evidence that a positive intervention at one stage does not insulate a person from getting derailed at a later point when a chilly climate is encountered.² So, if one middle school teacher adopts innovative approaches to engage girls in computing, her gains may be lost if the girls find no CS classes or male-dominated classes in high school. If one technology department adopts effective practices for mentoring diverse new hires, the gains may be lost for lack of mentors at mid-career. What is needed is widespread systemic change. Yet each existing BP organization is focused on one or a few underrepresented groups, pipeline stages, or geographical regions. No organization addresses the general issue of BP or the collective needs for people of all ages and in various settings. No organization convenes leaders of other BP organizations, or sits at the table with ACM SIG leaders, with goals to connect similar efforts, identify gaps, and strategically plan for the future. There is little international collaboration around BP.

Referencing. A central referencing mechanism is not available to enable

BP researchers and implementers to build upon existing work. Many smaller groups are undertaking independent grassroots approaches to create programs, with limited knowledge of similar work. No organization serves as a primary reference for BP research and practice in general. Though computing researchers are making important advances in BP, the literature is scattered among journals and conferences within computing, engineering, science, education, and the social sciences. Though a number of BP portals are emerging, only www.BPCPortal.org aims to serve all audiences, but falls short due to a lack of awareness of its existence.

Awareness. There continue to be students, educators, professionals, and policy makers who are unaware or remain unconvinced of the need for BP. While many initiatives are demonstrating “effective practices,” too few schools, colleges, and companies are integrating these practices into their core mission. As such, many existing BP activities are producing islands of change, but there is no unified effort to connect these islands and to ensure they are not washed away when a program’s funding ends or a program’s champion moves on. In some cases, this requires policy changes in organizations and at the state and federal level. No organization serves as the central voice for BP advocacy.

What is needed is an overarching community to connect, support, and amplify BP initiatives worldwide, and a computing publications venue to provide referencing to practices and research. The ACM SIGs are a well-recognized model for fostering international communities around computing related areas. A SIGBP would serve as a central resource for all of the BP efforts of other organizations. SIGBP meetings would be a gathering of the leadership of these organizations to coordinate and plan for the future. A SIGBP could serve to connect the islands of BP activity to magnify and accelerate the outcomes of these activities.

Proposed SIGBP Activities

The formation committee to create an ACM SIGBP was formed in 2009. Members include leaders of the ACM-W, CRA-W, ABI, GHC, CDC, Tapia Celebration, NSF, and NSF BPC Alli-

ances (STARS Alliance, Computing Alliance for Hispanic Serving Institutions, Access Computing, Alliance for African American Researchers in Computing, and NCWIT). Over 300 researchers, educators, and practitioners have signed a petition supporting the SIGBP; 75 people expressed interest in SIGBP leadership. The initial leadership team will include: Monica Anderson, Maureen Biggers, Brian Blake, Ann Condon, Juan Gilbert, Richard Ladner, Bobby Schnabel, George K. Thiruvathukal, Manual Perez Quinones, and myself. Endorsement letters have been received from 16 organizations, including ACM SIGCSE, ACM SIGITE, ABI, CDC, CRA-W, NCWIT, and the Natural Sciences and Engineering Research Council of Canada. We received NSF funding for SIGBP startup.^a The committee held several meetings and community forums to inform the vision of the SIGBP, as summarized next.

The scope of a SIGBP is to provide an overarching international community to highlight, support, connect, and strengthen the diverse array of existing BP communities, activities, and research. The mission is to advance innovation and discovery by increasing and broadening participation in computing. The SIG will promote research, technologies, practices, and policies for increasing the participation of women, underrepresented minorities and persons with disabilities in computing.

SIGBP members will be researchers, practitioners, educators, and policymakers from academia, industry, government, K-12 schools, and the non-profit sector within a variety of interests, including computing, psychology, sociology, ethnic and gender studies, education, and human resources.

The SIGBP goals are to foster collaboration, dissemination, and support among organizations, programs, and people seeking to broaden participation in computing, with a fourfold focus on research, practices, awareness, and professional development, as follows:

- Advance research on the factors behind lack of participation. The SIG-

The need to involve all voices to inform change and provide equitable access to jobs is urgent.

BP will provide greater opportunity for computing researchers to publish in a computing venue. This will provide a referencing mechanism to research with an explicit focus on BPC, research on broadening participation in science and engineering that is relevant for computing, and publications in the many disciplines that inform BPC research, such as education, psychology, gender studies, ethnic studies, educational technologies, student services, and human resources. SIGBP peer-reviewed publications will make the outcomes of BPC research more visible to computing researchers and practitioners who are not already connected to BPC activities.

- Evaluate and promulgate effective practices for increasing participation. The SIGBP will provide a central community to find information about organizations, activities, publications, awards, other communities and portals with a BP mission. Examples of BP practices to be disseminated include models, frameworks, and strategies for recruiting and advancing people along the pipeline from K-12 to college to the workforce.

- Advocate beneficial institutional or government policies. The SIGBP will raise awareness among policymakers and the public at large, by seeking to inform and engage the computing community in emerging efforts to address policy.

- The SIGBP will advance the academic and career development of students and professionals in computing by providing an index to the many opportunities to participate in programs with goals to broaden participation.

SIGBP activities will include:

- Sponsor an annual SIGBP symposium, co-located with an existing

conference, publishing peer-reviewed papers.

- Publish a newsletter that reports on recent activities, advertises events, highlights new results and articles, and includes descriptions of member organizations.

- Co-sponsor existing conferences, events, and publications that focus on BP or have a broader engagement component.

- Partner with other ACM SIGs to enhance and promote their broader engagement initiatives.

- Provide a repository of best practices, enhancing the library at bpcportal.org.

- Advertise opportunities for students offered within industry, government, academia, K-12 or community organizations seeking to broaden participation (for example, summer experiences, mentoring, conferences, scholarships).

- Provide a central portal to organizations, activities, publications, awards and other portals with a BP mission.

The SIGBP will collaborate with other related ACM SIGs, including SIGACCESS, SIGCSE, and SIGITE. The SIGBP might best be compared to the SIGCAS (Computers and Society) and SIGUCCS (University and College Computing Services), because of the multipoint focus that includes research, sharing best practices, professional development, and raising awareness.

Computing and technology are game-changers for the 21st century, impacting how we work and live. The need to involve all voices to inform change and provide equitable access to jobs is urgent. We cannot bring about systemic change through islands of activity. An ACM SIGBP will connect these islands to accelerate systemic change. Express your support at <http://bit.ly/sig-bp-form>. 

References

1. Chubin, D.E. and Johnson, R.Y. A program greater than the sum of its parts: The BPC alliances. *Commun. ACM* 54, 3 (Mar. 2011).
2. Etzkowitz, H., Kemelgor, C., and Uzzi, B. *Athena Unbound: The Advancement of Women in Science and Technology*. Cambridge University Press, 2000.
3. Ladner, L. Opening remarks. *Commun. ACM* 52, 12 (Dec. 2009).

Teresa A. Dahlberg (Teresa.Dahlberg@unc.edu) is a professor of computer science and Associate Dean of Undergraduate Programs and Administration for the College of Computing and Informatics at the University of North Carolina at Charlotte.

Copyright held by author.

^a NSF Award 1042372 (T. Dahlberg), 1042337 (G. Thiruvathukal), ACM SIGBP: Forming an ACM Special Interest Group to Scale the Impact of BPC Activities.

Viewpoint

Alan Turing Remembered

A unique firsthand account of formative experiences with Alan Turing.

IN 1945 MY father, Max Newman, who had been working at Bletchley Park during the Second World War, was appointed head of the Mathematics Department at Manchester University. At that time my parents, my brother Edward, and I were living at Cross Farm in the village of Comberton, near Cambridge. My father's new job required us to move close to Manchester, and my parents therefore bought a house in Bowdon, Cheshire. In 1946 we moved there from Cross Farm, which was rented out to tenants.

During the postwar years a steady stream of academics visited us at the Bowdon house. Some of them were bright young mathematicians who were being considered for teaching posts in the department. Others were more senior academics, invited by my father to give lectures: John Cockcroft, Mary Cartwright, Henry Whitehead, Norbert Wiener, and others. In those postwar years it would have been expensive for these visitors to stay in hotels, so they stayed with us, usually for just one night.

During the late 1940s our most frequent guest was Alan Turing. In 1946

my father, having secured a substantial grant from the Royal Society for computer research, had offered Alan a position as a Reader in the Mathematics Department, and Deputy Director of the Computing Laboratory. Alan accepted, taking up the position in 1948 and later buying a house ("Hollymeade") in Wilmslow, eight miles from Bowdon. Living so close, Alan was now able to visit us quite frequently on weekends. He usually made the journey on his bicycle, which had been fitted with a small gasoline engine. His visits were enjoyed by all



PHOTO ILLUSTRATION BY ALICIA KUBISTA

of us: by my father, who could sit with Alan and explore avenues for artificial intelligence research; by my mother, who was constantly touched by Alan's honesty and originality when they talked together; and by me and my brother, both of us happy to have a grown-up friend who liked spending time with us—and spending money on delightful presents for us on our birthdays and at Christmas.

One of Alan's visits to our Bowdon house, early one autumn morning in 1950, still gives me pleasure to recall. I was asleep in the ground-floor front room, and was awakened by a rustling noise at our front door. I went to open it, and found Alan standing there in his running clothes. He explained that he had set out to run from Wilmslow to Bowdon and back, and on the way had decided to invite us to dinner at his house. Arriving at our front door, he was concerned not to wake us. He had therefore used a twig to scratch his invitation on a leaf that he picked off our rhododendron bush.

I also recall several visits when Alan and I played games. On one occasion he suggested a game of “blind-fold” chess, during which he would sit where he could not see the board and would call out his moves, while I would sit with the board in front of me and call out mine. The game got under way, my father watching its progress in silence. After a while Alan excused himself and left the room. He was gone for a several minutes, during which time my father took the opportunity to point out that my king was in a precarious position. Eventually Alan—who must have realized this—returned and announced the winning move. I believe he had not wanted to upset me by winning, and had left the room to steel himself for issuing his *coup de grace*.

I remember a later phone call from Alan, asking me if Edward and I had a Monopoly set. I replied that we did, and was delighted when he suggested we three might play a game when he next visited. At the time it did not occur to me to mention that our “set” was homemade, with a “board” drawn by me on a sheet of paper. I could also have told him that its layout was somewhat unorthodox—I had added an extra row of properties, diagonally con-

Living so close, Alan was able to visit us quite frequently on weekends.

necting the “GO” square to the square on the opposite corner. I cannot recall why I had done this, but I may have wanted to provide a choice of routes. In the end Alan lost—neither Edward nor I showed him any mercy. [The original homemade Monopoly board was discovered last year in the Newmans' old family home and was recently released as a special edition: see <http://www.heritageandhistory.com/contents1a/2011/06/monopoly-board-secured-by-spy-hq/>. William Newman's original game-board drawing is on display at the Bletchley Park Museum.—*Ed.*]

It was at around this time that Alan inadvertently attracted the attention of the *Times*, by giving a talk about the computer research in Manchester. My mother described the event in a letter to a friend: “Did you see the extraordinary report in the *Times* two weeks ago on the Manchester Calculating Machine with the fantastic remarks attributed to Alan Turing? And Max's letter the following week trying to clear things up? The *Times* wired Alan, who isn't on the telephone, to ring their office, and they interviewed him on the phone. He's wildly innocent about the ways of reporters and has a bad stammer when he's nervous or puzzled. It was a great shock to him when he saw the *Times*—and to Max who had been flying back from Belfast that day. We had a wretched weekend starting at midnight on the Friday night when some sub-editor of a local paper rang up to get a story. By Sunday Max was getting a bit gruff, and when he said, ‘What do you want?’ to one newspaper, the reporter replied, ‘Only to photograph your brain.’”^a

a Lyn Newman to Antoinette Esher, June 24, 1949.

For my parents, moving to Manchester meant finding me a local school, but none of the local grammar schools taught mathematics to a standard that satisfied my father. Meanwhile, my mother was having trouble finding suitable tenants for Cross Farm, and after four years in Bowdon she was missing her Cambridge friends and her sister Yda in London. In April 1950 she therefore moved back to Cross Farm with me and Edward, and my father found an apartment in Bowdon where he could stay during term time. I was enrolled at the County grammar school in Cambridge, where I received extra tuition in mathematics from a genial Yorkshireman, Tom Marsden. My father approved of this choice of school, possibly because the headmaster, Brinley Newton-John, had worked at Bletchley Park.

Our move to Cross Farm separated us from Alan, and both my mother and I missed him greatly. She had become very fond of Alan over the years, and he had opened up to her. I have no memories of having any contact with Alan during the last four years of his life. Somehow my parents ensured I was unaware of his 1952 conviction for gross indecency. News of his death reached us at Cross Farm on June 8, 1954, the day after his body was discovered: the phone rang while my mother and I were in the living room, and she answered it. I could tell at once that the call was from my father and that the news was bad. My mother sat huddled over the phone, saying very little except to ask questions: Had Dr. Greenbaum, Alan's psychiatrist, been informed? She was in floods of tears when the call ended, and all she could then say to me was that Alan had taken his own life. Later that day she felt able to say a few more words. She told me that Alan was a homosexual, and explained what that term meant. A few days later she attended Alan's funeral, and handed the letters she had received from him to his brother John. A chapter in our lives had come to an end. 

William Newman (wmn@pobox.com) is a visiting professor at University College, London, U.K.

Copyright held by the author and Lord Esher.



27th

IEEE INTERNATIONAL PARALLEL & DISTRIBUTED PROCESSING SYMPOSIUM

www.ipdps.org



*Hyatt Regency Cambridge
Boston, Massachusetts USA
20 - 24 May 2013*

IPDPS 2013 will be held along the banks of the Charles River at the Hyatt Regency Cambridge with splendid views of the Boston skyline. The Boston-Cambridge venue offers an exciting multicultural setting in the midst of world renowned educational institutions and landmarks of America's founding, all navigable by foot and public transit. The IPDPS event offers a rich program of workshops, a PhD Forum, and the technical program of contributed papers, invited speakers, and tutorials.

GENERAL CO-CHAIRS

Martin Herbordt (Boston University, USA)
Charles C. (Chip) Weems (University of
Massachusetts at Amherst, USA)

PROGRAM CHAIR

Sanjay Ranka (University of Florida, USA)

WORKSHOPS CHAIR

Ümit V. Çatalyürek (Ohio State University, USA)

LOCAL PARTICIPATION & COMMERCIAL CHAIR

Ayşe Coskun (Boston University, USA)

IPDPS extends a special invitation to Boston area companies to consider participation in IPDPS 2013 to share in the benefits of associating with an international community of top researchers and practitioners in fields related to parallel processing and distributed computing. Visit the IPDPS Web for details.

PhD FORUM CO-CHAIRS

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

The PhD Forum 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. The deadline for submissions is January 14, 2013. See details at www.ipdps.org.

IPDPS 2013 WORKSHOPS

Held on the first and last day of the conference, workshops are a major part of the IPDPS week-long family of events, providing the IPDPS community an opportunity to explore special topics and present work that is more preliminary or cutting-edge 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. Note that the submission date for most workshops falls sometime after author notification date of December 23rd for regular conference papers.

HCW	Heterogeneity in Computing
RAW	Reconfigurable Architectures
HIPS	High-Level Parallel Programming Models & Supportive Environments
NIDISC	Nature Inspired Distributed Computing
HiCOMB	High Performance Computational Biology
APDCM	Advances in Parallel and Distributed Computing Models
CASS	Communication Architecture for Scalable Systems
HPPAC	High-Performance, Power-Aware Computing
HPGC	High-Performance Grid and Cloud Computing
ASHES	Accelerators and Hybrid Exascale Systems
STDN	Security and Trust of Distributed Networking Systems
EduPar	NSF/TCPP -Parallel and Distributed Computing Education
PDSEC	Parallel and Distributed Scientific and Engineering Computing
DPDNS	Dependable Parallel, Distributed and Network-Centric Systems
MTAAP	Multi-Threaded Architectures and Applications
LSPP	Large-Scale Parallel Processing
PCO	Parallel Computing and Optimization
ParLearning	Parallel & Distributed Computing for Machine Learning & 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
ViPES	Virtual Prototyping of Parallel and Embedded Systems

Sponsored by

IEEE  computer society
Technical Committee on Parallel Processing

In cooperation with

 ACM SIGARCH
 IEEE Computer Society Technical Committee on Computer Architecture
 IEEE Computer Society Technical Committee on Distributed Processing

A thinking framework in the form of an actionable kernel.

BY IVAR JACOBSON, PAN-WEI NG, PAUL E. MCMAHON, IAN SPENCE, AND SVANTE LIDMAN

The Essence of Software Engineering: The SEMAT Kernel

EVERYONE WHO DEVELOPS software knows it is a complex and risky business, and its participants are always on the lookout for new ideas that will lead to better software. Fortunately, software engineering is still a young and growing profession that sees innovations and improvements in best practices every year. Just look, for example, at the improvements and benefits that lean and agile thinking have brought to software-development teams.

Successful software-development teams need to strike a balance between quickly delivering working software systems, satisfying their stakeholders,

addressing their risks, and improving their ways of working. For that, they need an effective thinking framework that bridges the gap between their current way of working and any new ideas they want to adopt. This article presents such a thinking framework in the form of an actionable kernel, which could benefit any team wishing to balance their risks and improve their way of working.

Work on the kernel, the essence of software engineering, was inspired by and is a direct response to the Software Engineering Methods and Theory (SEMAT) call for action (see Figure 1). It is, in its own way, one small step toward redefining software engineering.

SEMAT was founded in September 2009 by Ivar Jacobson, Bertrand Meyer, and Richard Soley, who felt the time had come to fundamentally change the way people work with software-development methods.^{3,4,8} They wrote a call for action statement, which in a few lines identifies a number of critical problems, explains why there is a need to act, and suggests what needs to be done. The call for action is:

Some areas of software engineering today suffer from immature practices. Specific problems include:

- ▶ The prevalence of fads more typical of the fashion industry than an engineering discipline;
- ▶ The lack of a sound, widely accepted theoretical basis;
- ▶ The huge number of methods and method variants, with differences little understood and artificially magnified;
- ▶ The lack of credible experimental evaluation and validation; and
- ▶ The split between industry practice and academic research.

The SEMAT call for action's assertion that the software industry is prone to fads and fashions has led some people to assume that SEMAT is resistant to new ideas. This could not be further from the truth. As you will see in this article and in a soon-to-be-published book (*The Essence of Software Engineering—Applying the SEMAT Kernel*),⁶ SEMAT supporters are very keen on new

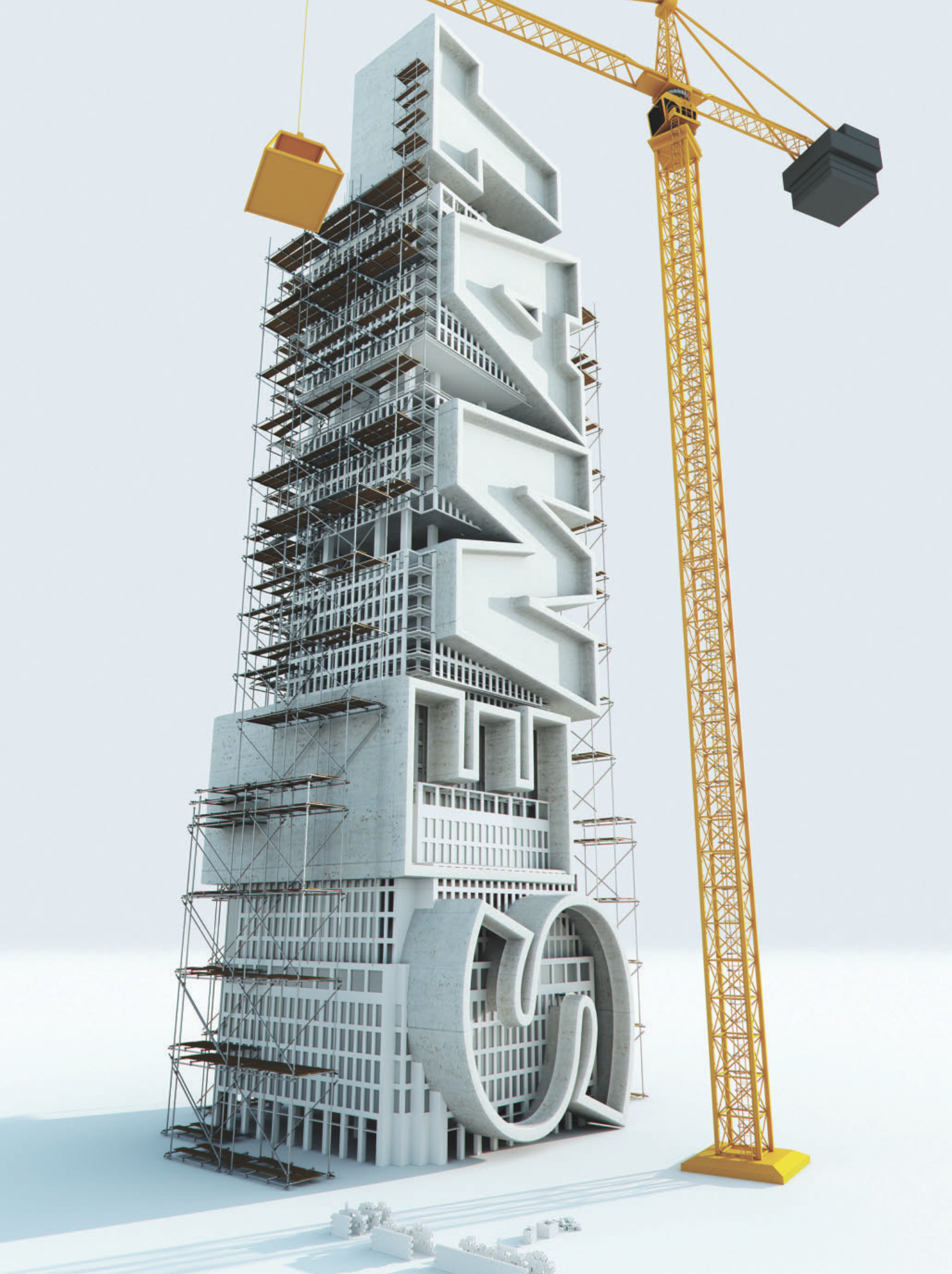


Figure 1. Things to work with.

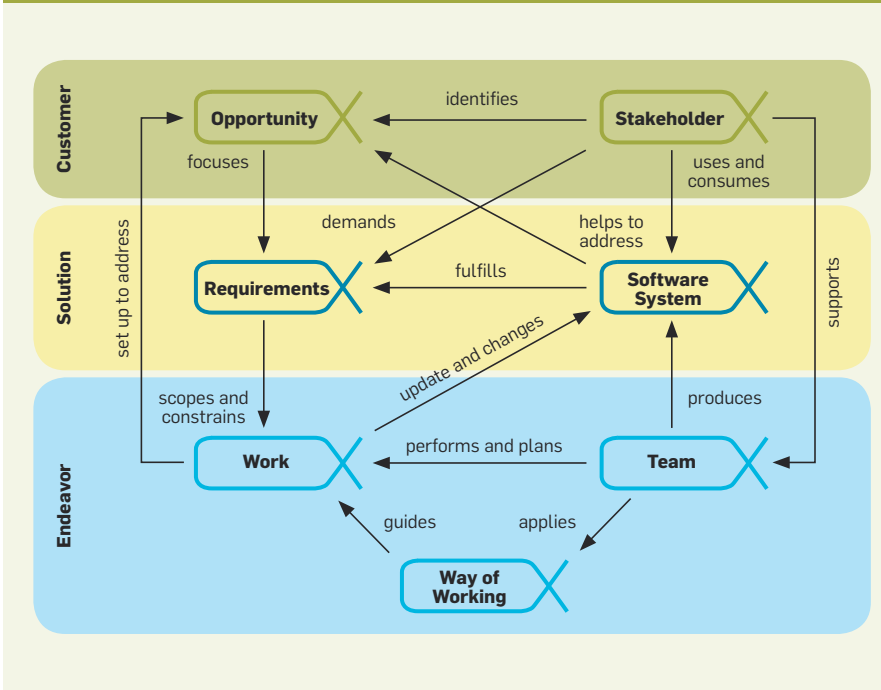
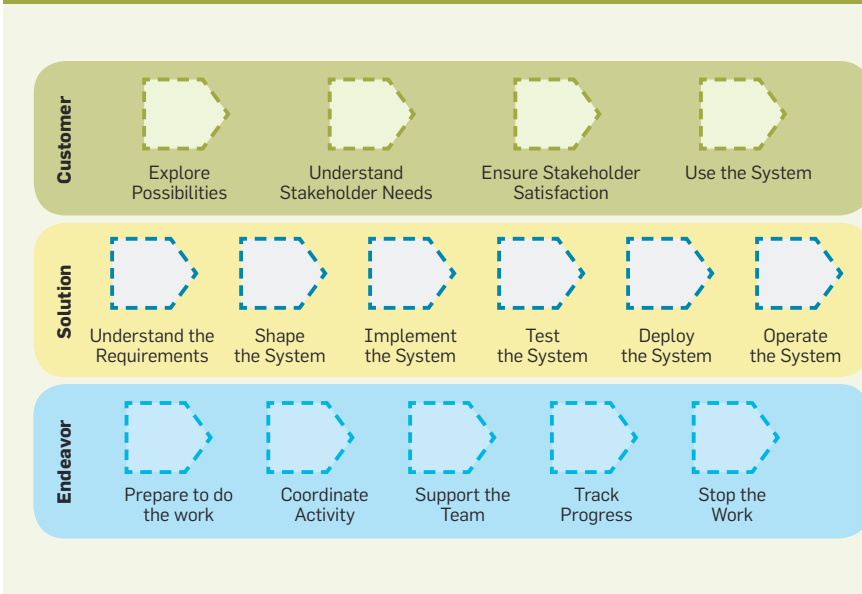


Figure 2. Things to do.



ideas. What they are against is the non-lean, non-agile behavior that comes from people adopting inappropriate solutions just because they believe these solutions are fashionable—or because of peer pressure or political correctness.

SEMAT supports a process to redefine software engineering based on a solid theory, proven principles, and best practices that:

- Include a kernel of widely agreed-upon elements, extensible for specific uses;

- Address both technology and people issues;

- Are supported by industry, academia, researchers, and users; and

- Support extension in the face of changing requirements and technology.

The SEMAT call for action received a broad base of support, including a growing list of signatories and supporters (<http://www.semat.org>). In February 2010 the SEMAT founders developed the call for action into a vision statement.⁵ In accordance with

this vision, SEMAT focused on two major goals: Finding a kernel of widely agreed-upon elements, and defining a solid theoretical basis.

To a large extent these two tasks are independent of each other. Finding the kernel and its elements is a pragmatic exercise requiring experienced software developers with knowledge of many of the existing methods. Defining the theoretical basis is academic research and may take many years to reach a successful outcome.

The Power of the Common Ground

SEMAT's first step was to identify a common ground for software engineering. This common ground is manifested as a kernel of essential elements that are universal to all software-development efforts, and a simple language for describing methods and practices. The kernel was first published in the SEMAT OMG (Object Management Group) submission.^{2,9} As shown in figures 1 and 2, the kernel contains a small number of “things we always work with” and “things we always do” when developing software systems. Work is also ongoing to define the “skills we always need to have,” but this will have to wait until future versions of the kernel.

More than just a conceptual model, the kernel provides:

- A thinking framework for teams to reason about the progress they are making and the health of their endeavors;

- Common ground for the discussion, improvement, comparison, and sharing of software-engineering methods and practices;

- A framework for teams to assemble and continuously improve their way of working by the composition of separately defined, and sourced, practices;

- A foundation for defining practice-independent measures to assess the quality of the software produced and the methods used to produce it; and

- Most importantly, a way of helping teams understand where they are, what they should do next, and where they need to improve.

The Big Idea

What is it that makes the kernel more than just a conceptual model of software engineering? What is it that is re-

ally new here? This can be summed up in its founding principles (see Figure 3), which really bring out three unique features of the kernel: it is actionable; it is extensible; and it is practical.

The kernel is actionable. A unique feature of the kernel is the way that it handles the “things to work with.” These are captured as *alphas* rather than work products (such as documents). An alpha is an essential element of the software-engineering endeavor—one that is relevant to an assessment of its progress and health. As shown in Figure 1, SEMAT has identified seven alphas: opportunity, stakeholders, requirements, software system, work, way of working, and team.

The alphas are characterized by a simple set of states that represent their progress and health. As an example, the software system moves through the states of architecture selected: demonstrable, usable, ready, operational, and retired. Each state has a checklist that specifies the criteria needed to reach the state. These states make the kernel actionable and enable it to guide the behavior of software-development teams.

The kernel presents software development not as a linear process but as a network of collaborating elements that need to be balanced and maintained so teams can make effective and efficient progress, eliminate waste, and develop great software. The alphas in the kernel provide an overall framework for driving and progressing software-development efforts, regardless of the practices applied or the philosophy followed.

As practices are added to the kernel, alphas will be added to represent the things that either drive the progress of the kernel alphas or inhibit their progress. For example, the requirements alpha will not be addressed as a whole but will move forward item by item. The progress of the individual requirement items will either drive or inhibit the progress and health of the requirements alpha. The requirements items could be of many different types: for example, features, user stories, or use-case slices, all of which can be represented as alphas and have their states tracked. The benefit of relating these smaller items to the coarser-grained kernel elements is that it allows the tracking of the health of the endeavor as a whole. This provides a necessary balance to the lower-level tracking of the individual items, enabling teams to understand and optimize their ways of working.

The kernel is extensible. Another unique feature of the kernel is the way it can be extended to support different projects (for example, new development, legacy enhancements, in-house development, offshore development, software product lines, and so on). The kernel allows you to add practices, such as user stories, use cases, component-based development, architecture, pair-programming, daily stand-up meetings, self-organizing teams, and so on to build the methods you need. For example, different methods could be assembled for in-house and outsourced development or for the development of safety-critical embedded systems and back-office reporting systems.

Practice separation is the key idea here. While the term *practice* has been widely used in the industry for many years, the kernel has a specific approach to the handling and sharing of practices. Practices are presented as distinct, separate, modular units, which a team can choose to use or not to use. This contrasts with traditional approaches that treat software development as a soup of indistinguishable practices and lead teams to dump the good with the bad when they move from one method to another.

The kernel is practical. Perhaps the most important feature of the kernel is the way it is used in practice. Traditional approaches to software-development methods tend to focus on supporting process engineers or quality engineers. The kernel, in contrast, is a hands-on, tangible thinking framework focused on supporting software professionals as they carry out their work.

For example, the kernel can be touched and used through cards (see Figure 4).^{7,10} The cards provide concise reminders and cues for team members as they go about their daily tasks. By providing practical checklists and prompts, as opposed to conceptual discussions, the kernel becomes something the team uses on a daily basis. This is a fundamental difference from traditional approaches, which tend to overemphasize method description as opposed to method use and tend to be consulted only by people new to the team.

Cards provide concise descriptions that serve as reminders for team members. They can keep the kernel as

Figure 3. Guiding principles behind the kernel.

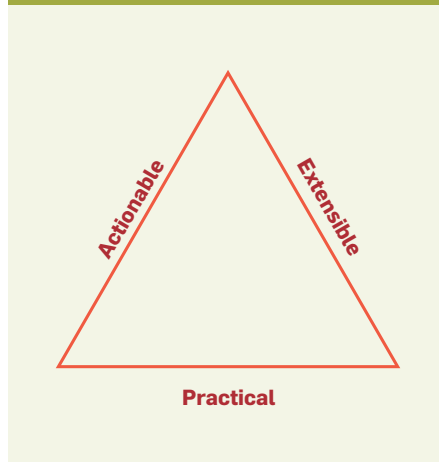


Figure 4. Cards make the kernel tangible.



a small deck of cards in their pockets, which they can easily pull out to discuss the current status of development and the work assignment and collaboration among team members. Teams can also discuss areas of improvement by referring to the cards. Thus, the kernel is not merely a heavyweight description of what a team needs to do. Rather, it forms an essential part of what they are doing each day.

The kernel in action. The kernel has many applications in software professionals' everyday lives. They include:

- ▶ Running an iteration (or sprint).
- ▶ Running the entire development from idea to product.
- ▶ Scaling to large organizations and complex software-development endeavors.

The first application, planning an iteration, is used here as an example of what a team can do with the kernel. The others are covered fully in *The Essence of Software Engineering—Applying the SEMAT Kernel*.⁶

The example presented here assumes that a company has very little

in the way of formal processes. In the past it has relied on having skilled and creative individuals on experienced teams, but the company is now growing and has many new hires. These new employees, mostly fresh out of university, have good technical skills—for example, in programming languages—but are less equipped in other aspects of software development, such as working with stakeholders to gain agreement on requirements.

This company has a development team that is responsible for making a mobile social-network application that lets users share and browse ideas, photos, and comments. The team began with only two developers, Smith (the team leader) and Tom, both of whom are familiar with the kernel. They are later joined by two more developers, Dick and Harriet, who are new to the job and have no previous knowledge of the kernel. Success to team leader Smith means more than functionality, schedule, and quality. This team ran development iteratively. You can think of planning an iteration as follows:

1. Determine where you are: Work out the current state of the endeavor.

2. Determine where to go: Decide what to emphasize next and what the objectives of the next iteration will be.

3. Determine how to get there: Agree on the tasks the team needs to do to achieve the objectives.

Determine Where the Team Is Using the Kernel

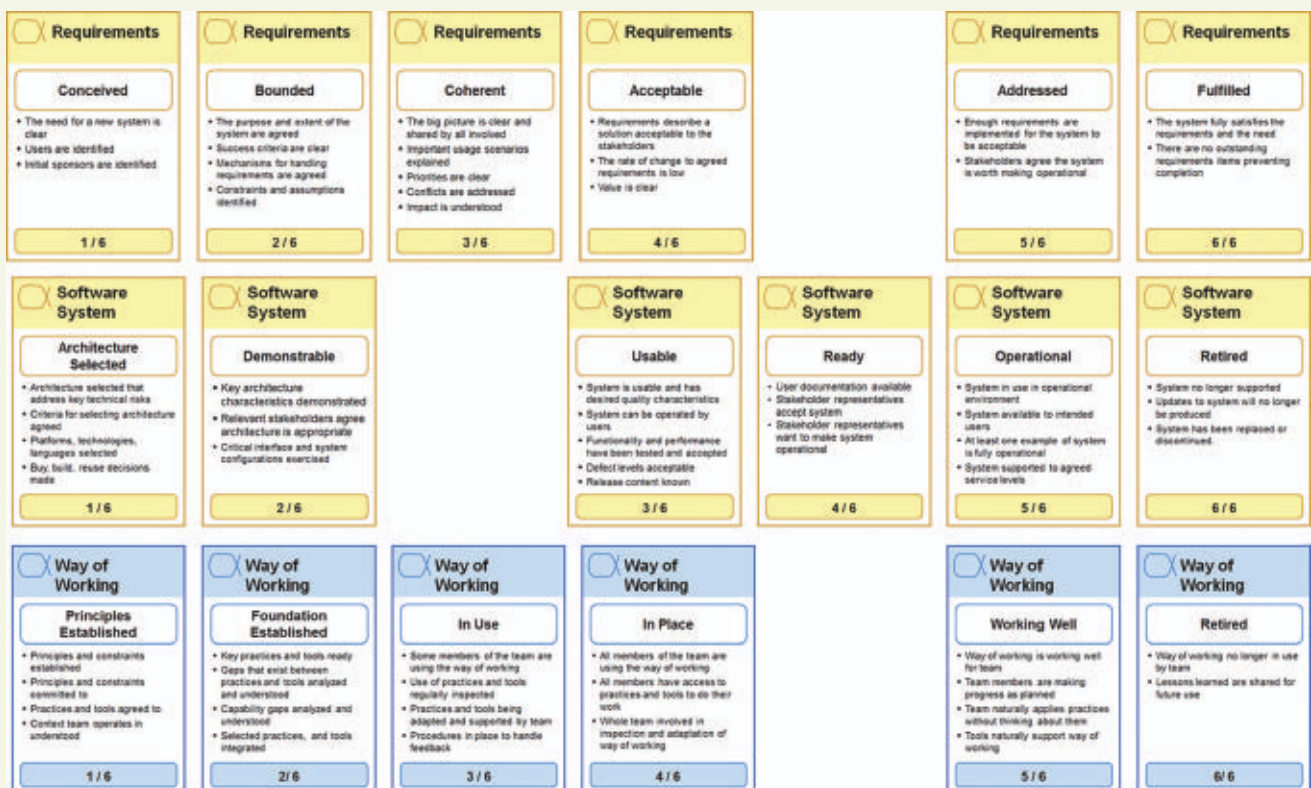
Let's assume that Smith and his team are six weeks into development. They have provided an early demonstration of the system to their stakeholders, who are pleased and provide valuable feedback. The system is not yet usable by end users, however. You can use the kernel to do this in a number of ways. If you are using alpha state cards, then you can do a walk-through as follows:

1. Lay out the cards for each alpha in a row on a table with the first state on the left and the final state on the right.

2. Walk through each state and ask your team if you have achieved that state.

3. If the state is achieved, move that

Figure 5. The team uses the alphas to determine the current states.



state card to the left. Continue with the next card until you get to the state that your team has not yet achieved.

4. Move this state card and the rest of the pending state cards to the right.

Figure 5 shows the states Smith's team has achieved on the left, and those not yet achieved on the right. For simplicity, Figure 5 shows only three of the kernel alphas.

Determine where to go with the kernel. Once the team agrees on the current alpha states, the members discuss what the next desired "target" states are to guide their planning. The team agrees to use the immediate next alpha states to help establish the objectives of the next iteration, as shown in Figure 6.

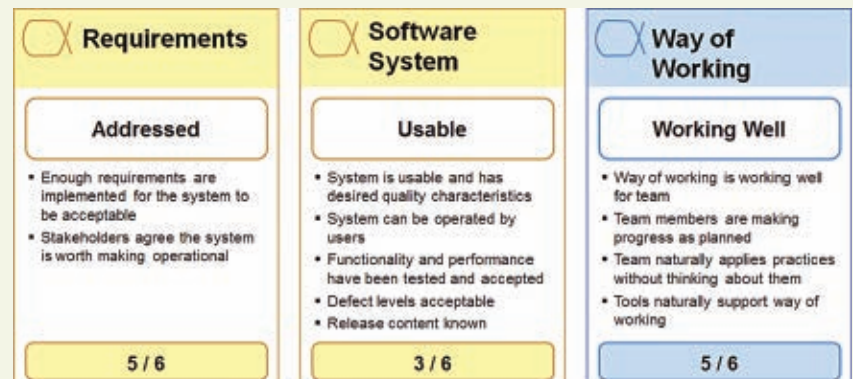
The name of the alpha state supplies a hint to understanding what needs to be achieved to reach a state. Team members can find out more by reading and understanding the alpha-state checklist. By going through the states one by one for each alpha, a team quickly becomes familiar with what is required to achieve each state. In this way the team learns about the kernel alphas at the same time as they determine their current state of development and their next target states.

Determine how to get there with the kernel. Smith and his team look at the next target states and agree that they need to establish some priorities. First, they need to determine their *way of working: working well*; then the *software system: usable*; and finally *requirements: addressed*. The reason is simple; not having the way of working: working well would impede their attempts to get the *software system: usable*. In addition, they agree on the priority for the missing requirement-items necessary to achieve the *requirements: addressed* state.

Smith and his team next discuss what needs to be done to achieve these states, as shown in the accompanying table. By going through the target alpha states, Smith is able to determine a set of objectives and tasks for the next iteration.

How the kernel helps in planning iterations. A good plan must be *inclusive*, meaning that it includes all essential items and covers the whole team. It must also be *concrete*, so it is actionable for the team. The team must also have

Figure 6. The selected next steps.



How the team achieves states.

Target State

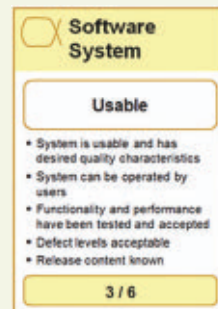
How they planned to achieve them



Both Dick and Harriet agreed that they had difficulties in applying automated testing. They needed help in order to make progress. Tom agreed that he had to spend time teaching them:

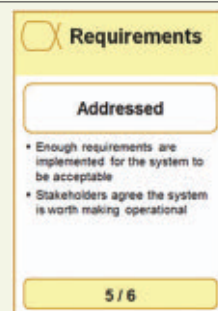
A task was added to the iteration backlog for Tom to conduct training on automated testing for Dick and Harriet.

- Task 1. Conduct training on automated testing.



This state reminds us that the software system must be shown to be of sufficient quality and functionality to be useful to the users. So far, Smith's team had been testing within its development environment. Now, it had to conduct tests within an acceptance-test environment, which they had yet to prepare. This resulted in the following task:

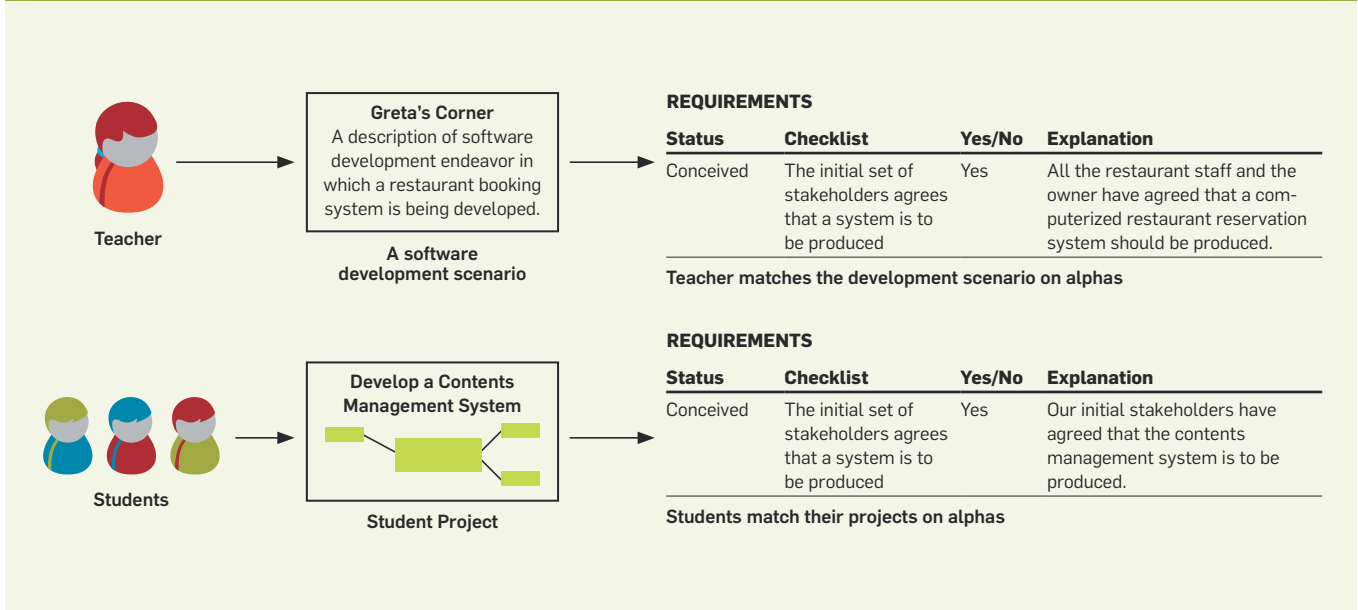
- Task 2. Prepare acceptance-test environment. Smith's team had to bring all requirement-items currently demonstrable in the system to completion. This meant that each requirement-item must be fully tested within the acceptance-test environment.
- Task 3. Complete requirement-item A: "Browse online and offline."
- Task 4. Complete requirement-item B: "Post comment (online and offline)."
- Task 5. Complete requirement-item C: "Browse album."



This state reminds us of the need to work with stakeholders to ensure they are happy with the system produced. In our story Smith had to work with Angela, the customer representative, to determine which additional requirement-items needed to be implemented. This resulted in the additional task:

- Tasks 6: Talk to Angela and agree on additional requirements-items, fitting in the iteration, to make the system worth being operational.

Figure 7. Visualizing the pedagogical approach when teaching the SEMAT kernel.



a way to monitor its progress against the plan. The kernel helps you achieve this as follows:

Inclusive. The kernel alphas serve as reminders across the different dimensions of software development, helping create a plan that addresses all dimensions in a balanced way.

Concrete. The checklists for each alpha state hint at what you need to do in the iteration. The same checklists help determine your progress by making clear what you have done and comparing this with what you intended to do.

The Kernel in the Real World

Although the ideas presented here will be new to many of you, they have already been successfully applied in the real world by both industry and academia. In all cases they used the kernel and practices developed by Ivar Jacobson International.^{1,10} Early adopters of the kernel idea include:

- MunichRe, the world's leading re-insurance company, where a family of "collaboration models" has been assembled to cover the whole spectrum of software and application work. Four collaboration models—exploratory, standard, maintenance, and support—have been built on the same kernel from the same set of 12 practices.

- Fujitsu Services, where the Apt Toolkit has been built on top of an early version of the software-engineering kernel, including both agile and waterfall ways of working.¹

- A major Japanese consumer electronics company, where the software processes have been defined on top of an early version of the kernel, helping teams apply new practices and manage an offshore development vendor.

- KPN, where a kernel-based process was adopted by more than 300 projects across 13 programs as part of a move to iterative development. The kernel also provided the basis for a new results-focused QA process, which could be applied to all projects regardless of the method or practices used.

- A major U.K. government department, where a kernel-based agile toolset was introduced to enable disciplined agility and the tracking of project progress and health in a practice-independent fashion.

The kernel is already being used in first- and second-year software-engineering courses at KTH Royal Institute of Technology in Sweden. After students in the first-year courses conducted their projects, they went through the SEMAT alphas and matched them to their project results, under the direction of Anders Sjögren. The students had the opportunity to acquaint themselves with and evaluate the alphas and gain insight into the project's progress and health. In the second-year courses, run by Mira Kajko-Mattsson, the students were asked to use the SEMAT kernel when running their projects along with the development method they followed. As shown in Figure 7, Kajko-

Mattsson created a software-development scenario and evaluated it for each alpha, its states, and the state checklist items. The students were then asked to do the same when conducting and evaluating their projects.

The experiences of these courses provided valuable lessons. For example, the kernel assures that all the essential aspects of software engineering are considered in a project. By matching the project results against the kernel alphas, the students could easily identify the good and bad sides of their development methods. The kernel also prepared students for future software-engineering endeavors with minimal teaching effort. By following all the kernel alphas, the students could learn the total scope of the software-engineering endeavor and thereby see what would be required of them in their future as professionals.

How the kernel relates to agile and others. The kernel can be used with all the popular management and technical practices, including Scrum, Kanban, risk-driven iterative, waterfall, use-case-driven development, acceptance-test-driven development, continuous integration, and test-driven development. It will help teams embarking on the development of new and innovative software products and those involved in enhancing and maintaining established software products. It will help all sizes of teams from one-man bands to 1,000-strong software-

engineering programs.

For example, the kernel supports the values of the Agile Manifesto. With its focus on checklists and results, and its inherent practice independence, it values individuals and interactions over processes and tools. With its focus on the needs of professional software-development teams, it values the way of working and fulfilling team responsibilities over methods.

The kernel does not in any way compete with existing methods, be they agile or anything else. On the contrary, the kernel is agnostic to a team's chosen method. Even if a team is already using a particular method, the kernel can still help. Regardless of the method used, as Robert Martin pointed out in his foreword to *The Essence of Software Engineering*, projects—even agile ones—can get out of kilter, and when they do, teams need to know more. This is where the real value of the kernel lies. It can guide a team in the actions they need to take to get back on course, to extend their method, or to address a critical gap in their way of working. It focuses on the needs of the software professional and values the “use of methods” over “the description of method definitions” (as has been normal in the past).

The kernel does not just support modern best practices; it also recognizes that a vast amount of software is already developed and needs to be maintained. It will live for decades and will have to be maintained in an efficient way. This means the way you work with this software will have to evolve alongside the software itself. New practices will need to be introduced in a way that complements the ones already in use. The kernel provides the mechanisms to migrate legacy methods from monolithic waterfall approaches to more modern agile ones and beyond, in an evolutionary way. It allows you to change your legacy methods practice-by-practice, while maintaining and improving the teams' ability to deliver.

How the kernel will help you. Use of the kernel has many benefits for experienced or aspiring software professionals, and for the teams they work in. For example, it helps you assess the progress and health of software-development endeavors, evaluate current practices, and improve your way

of working. It also helps you improve communication, move more easily between teams, and adopt new ideas. It will help the industry as a whole by improving interoperability among teams, suppliers, and development organizations.

By providing a practice-independent foundation for the definition of software methods, the kernel also has the power completely to transform the way that methods are defined and practices are shared. For example, by allowing teams to mix and match practices from different sources to build and improve their way of working, the kernel addresses two of the key methodological problems facing the industry:

- Teams are no longer trapped by their methods; they can continuously improve their way of working by adding or removing practices when the situation demands.

- Methodologists no longer need to waste time describing complete methods; they can easily describe their new ideas in a concise and reusable way.

Finally, the kernel benefits academia. The kernel provides a basis for the creation of foundation courses in software engineering that can then be complemented with additional courses in specific practices—either as part of the initial educational curriculum or later during the student's professional development. Equally as important is the kernel's ability to act as a shared reference model and enabler for further research and experimentation. 

Related articles on queue.acm.org

There's No Such Thing as a Free (Software) Lunch

Jay Michaelson

<http://queue.acm.org/detail.cfm?id=1005066>

Purpose-Built Languages

Mike Shapiro

<http://queue.acm.org/detail.cfm?id=1508217>

Open Source to the Core

Jordan Hubbard

<http://queue.acm.org/detail.cfm?id=1005064>

References

1. Azoff, M. Apt Methods and Tools, Fujitsu. Ovum Technology Report. Reference Code 0100032-002 (Jan. 2011).
2. Fujitsu, Ivar Jacobson International AB, Model Driven Solutions. Essence—kernel and language for software engineering. Initial submission, 2012, version 1.0.
3. Jacobson, I. and Meyer, B. Methods need theory. *Dr. Dobb's Journal*, (2009).
4. Jacobson, I., Meyer, B. and Soley, R. Call for action:

The SEMAT initiative. *Dr. Dobb's Journal*, (2009).

5. Jacobson, I., Meyer, B. and Soley, R. The SEMAT vision statement, (2009).
6. Jacobson, I., Pan-Wei Ng, McMahon, P., Spence, I. and Lidman S. *The Essence of Software Engineering—Applying the SEMAT Kernel*. Addison-Wesley. (Forthcoming in Jan. 2013 but available in a prepublication version on safaribooksonline.com.)
7. Jacobson, I., Pan-Wei Ng and Spence, I. Enough of process—let's do practices. *Journal of Object Technology* 6, 6 (2007), 41-67.
8. Jacobson, I. and Spence, I. Why we need a theory for software engineering. *Dr. Dobb's Journal*, (2009).
9. Object Management Group (OMG). RFP: A foundation for the Agile creation and enactment of software engineering methods, (2012).
10. Pan-Wei Ng and Magee, M. Lightweight application lifecycle management using state cards. *Agile Journal* (Oct. 2010)

Ivar Jacobson, the chairman of Ivar Jacobson International, is a father of components and component architecture, use cases, the Unified Modeling Language, and the Rational Unified Process. He has contributed to modern business modeling and aspect-oriented software development.

Pan-Wei Ng coaches large-scale systems development involving many millions of lines of code and hundreds of people per release, helping them transition to a lean and agile way of working, not forgetting to improve their code and architecture and to test through use cases and aspects. He is the coauthor, with Ivar Jacobson, of *Aspect-oriented Software Development with Use Cases*.

Paul McMahon is an independent consultant focusing on coaching project managers, team leaders, and software professionals in the practical use of lean and agile techniques in constrained environments. He has been a leader in the SEMAT initiative since its inception.

Ian Spence is CTO at Ivar Jacobson International and the team leader for the development of the SEMAT kernel. He has introduced hundreds of projects to iterative and agile practices as well as led numerous successful large-scale transformation projects working with development organizations of up to 5,000 people.

Svante Lidman has extensive experience building high-performance enterprise software-development teams. He has held positions at Hansoft AB, Ivar Jacobson International, Microsoft, Rational Software, Objectory, among others. Since mid-2010 Lidman was the leading change agent in the largest lean/agile transition ever done in Scandinavia.

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

A proposal to improve the performance and availability of streaming video and other time-sensitive media.

BY AIMAN ERBAD AND CHARLES “BUCK” KRASIC

Sender-side Buffers and the Case for Multimedia Adaptation

THE INTERNET/WEB ARCHITECTURE has developed to the point where it is common for the most popular sites to operate at a virtually unlimited scale, and many sites now cater to hundreds of millions of unique users. Performance and availability are generally essential to attract and sustain such user-bases. As such, the network and server infrastructure plays a critical role in the fierce competition for users. Web pages should load in tens to a few hundred milliseconds at most. Similarly, sites strive to maintain *multiple nines* availability targets—for example, a site should be available to users 99.999% of the time over a one-year period.

Such stringent standards, however, are in practice carefully constructed to exclude problematic content such as video streaming and interactive virtual environments. If sites counted video-streaming rebuffer events against their availability, their multiple nines availability would vanish in a poof. Yet, one could argue that a video rebuffer just at the moment of the winning goal in the World Cup is a form of unavailability just as severe as, say, a failure that causes a social-network user to repeat some steps in composing a post. The goal of this article is to propose methods for adaptive transport that are ultimately aimed at bringing the performance and availability of streaming video and other time-sensitive media in line with that of traditional Web content. To do so we have developed an enhanced transport called Paceline.

IMAGE BY THOMAS PAJOT



As high-bandwidth low-latency multimedia applications such as videoconferencing, online games, and virtual reality applications become mainstream on the Internet, they will saturate the network, especially in mobile environments, much as other high-bandwidth (but high-latency) applications such as peer-to-peer file sharing do now. Videoconferencing, for example, is becoming a common service provided by social-networking sites (for example, Hangouts in Google). More than ever, real-time video streaming is becoming an integral component in many applications, such as live broadcasting of big events (for example, the Olympics, Super Bowl, and World Cup), distance learning, and on-demand video (Netflix). In addition, the global videogames industry was a \$64 billion business in 2012, growing larger than the movie industry. Fast-paced large-scale

games have high bandwidth requirements,¹ so they do not adhere to the old wisdom of network games having thin communication streams.

The combination of high bandwidth and low end-to-end latency is poorly supported in popular transports.^{3,8} Paceline is an enhanced transport designed to support interactive, high-bandwidth applications. Contrary to conventional wisdom in multimedia transports, Paceline has not been implemented over User Datagram Protocol (UDP), nor does it propose changes to TCP. The deployment obstacles and duplication of effort faced by solutions that alter or replace TCP outweigh the challenges of mitigating its impairments. Instead, Paceline uses several innovative techniques to address TCP latency problems.

On top of TCP transport delays, large sender-side application buffers

can accumulate and delay important data, exerting more influence over perceived quality. To have graceful failure modes for multimedia content in diverse environments (from gigabit broadband networks to congested wireless links), Paceline enables applications to adapt demands based on the available resources, and it favors important data. It supports quality adaptation based on the Priority-Progress model,⁹ shown to be more stable in terms of packet delay and jitter for video streaming over TCP.¹⁰

End-to-End TCP Latency Analysis

Since interactivity and transport latency are a key focus here, let's look at the sources of TCP latency and set the context for Paceline. As depicted in Figure 1, end-to-end transport latency is commonly broken down into four components: *processing* delay, as

a result of processing speed; *queuing* delays in nodes (hosts and network routers and switches); *transmission* delay resulting from the bit-rate of transmission; and *propagation* delays caused by physical distances. When one or more of those delays becomes large, interactivity (application-to-application message delivery) will suffer. Of these four latency components, *queuing delay* (inside TCP send buffers and network node queues) is the dominant cause of latency for high-bandwidth TCP applications. This is known as the bufferbloat problem.⁷

Processing delay is generally negligible because of fast CPUs and careful design of transport algorithms. Transmission delay will be bounded to

$$\text{delay}_{\text{transmit}} = \text{mss} / \text{link_rate}$$

assuming for the moment that ADUs (application data units) fit within transport segments up to an mss (maximum segment size). With common values of link_rate (Mbps or Gbps) and mss (for example, 1,500 byte), $\text{delay}_{\text{transmit}}$ will be a small value (for example, sub-millisecond). This leaves propagation delay and queuing delays as the dominant contributors to latency.

One-way propagation delay has lower bounds set by the laws of physics. Typical Internet path RTT (round-trip-time) values are in tens of milliseconds for intra-continental distances, or around 100 or 200 milliseconds for distances that cross oceans or traverse

satellites. In addition, TCP provides reliability via retransmissions that can add extra queuing delay (multiples of the propagation delay) to the total. In the common case, however, TCP's fast retransmit mechanism should limit the retransmission-induced queuing delay to an RTT or two.

More importantly, TCP's socket buffer is often large enough that it can cause queuing delays in seconds. In many realistic conditions, the queuing delay specifically caused by the sender-side TCP socket buffer is the dominant portion of the total delay because of large kernel socket buffers employed by TCP implementations. For example, with a typical TCP send buffer size of 64KB, and a 300Kbps video stream, a full send buffer contributes 1,700ms of delay. To avoid unnecessary queuing delays, the kernel can be changed to dynamically tune the socket buffer size, bringing the end-to-end delay within two RTTs most of the time, while leaving TCP's congestion control unchanged.⁸

Paceline builds upon this idea, but is designed to avoid the need for kernel modifications. A user-level approach avoids the deployment obstacles of introducing new TCP implementations, deals gracefully with transparent proxies that can defeat an in-TCP-based approach, and allows a failover mechanism to reduce the worst-case latency when TCP becomes stalled in the case of back-to-back losses and retransmission timeouts.

Data Service Model: Not All Data Is Born Equal

In diverse environments, demands often exceed available bandwidth, leading to large sender-side queues. Queues can introduce head-of-line blocking (a delay that occurs when a line of packets is held up by the first packet) and hinder perceived quality if all the data items are treated equally and processed in a FIFO (first-in first-out) order. Minimizing the amount of data committed to TCP socket buffers reduces TCP sender-side queuing delays and pushes the sender-side buffers up the stack to the layer above TCP (Paceline in our design). Here, we describe the transport service model in Paceline with the necessary quality adaptation mechanisms to manage

Figure 1. Components of end-to-end latency.

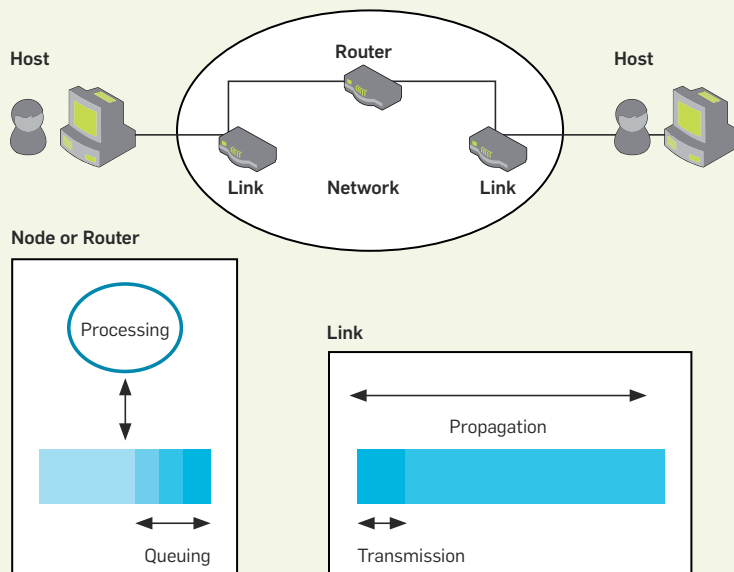
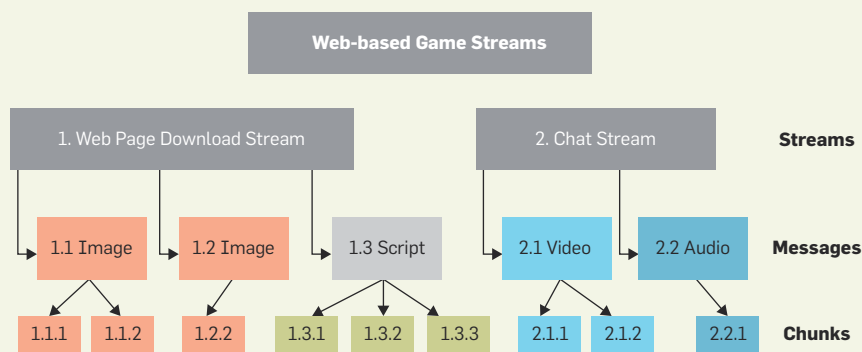


Figure 2. Streams of application data units.



the sender-side buffers based on the Priority-Progress model.

Paceline provides a reliable message-based service model, chosen because low latency is a primary goal and messages provide a natural explicit means for the application to inform the transport about latency preferences, as well as representing an ADU. Paceline's programming interface allows the application to specify message importance on a per-message basis, and Paceline delivers messages in order of importance. The ability to queue messages ahead of time is essential to achieve high bandwidth, but the ability to prioritize messages is necessary to prevent head-of-line blocking between messages of different importance levels and the attendant loss of responsiveness.

Unlike the byte-stream service model, Paceline allows the sender to cancel a pending message. This feature is motivated by the goal of responsiveness because the old data will slow down new messages and waste bandwidth. In conjunction with congestion control, cancellation is used by the application to adapt the rate of message delivery to the underlying network conditions. Informed cancellation maintains reliable delivery semantics while allowing applications to cancel stale messages. This provides an alternative to random dropping of messages (for example, UDP) under congestion. At the receiver, Paceline passes messages directly to the application. The application needs to handle out-of-order delivery and missing data introduced by message priority and cancellation.

In Figure 2, each ADU such as a video frame or a Web image uses a Paceline message. Each message is part of a full-duplex transport instance referred to as a *stream* (for example, video stream or Web document download stream). Similar to SPDY,² an application-layer transport protocol designed for minimal latency, Paceline supports multiplexing, which multiplexes concurrent streams on top of a single TCP channel. Applications can perform the following operations on streams: creation, sending a message, cancelling a message, and deletion. Streams are decoupled from the underlying TCP channels since all streams with the same host address and port num-

ber are multiplexed over the same TCP channel. A *channel* is the underlying communication primitive, identified by the host address and port number.

Like SPDY, Paceline efficiently multiplexes small Web transactions on top of a single TCP channel using short-lived streams (that is, those with few messages). More importantly, its innovative fairness design allows timely communication across concurrent long-lived streams such as videoconferencing or games. Different high-bandwidth streams can share a link and have varying requirements in terms of latency and high-level application-quality metrics (for example, frame rate). For example, a game transfers several kinds of streams such as player status updates, player video coordination chats, advertisements, and game control messages. The frequency of advertisements might be relaxed if necessary to help ensure player updates are sent promptly. Similarly, a distance-learning session can transfer voice, video, and slides from different users as separate streams multiplexed over the same TCP channel and can have different quality metrics.

To help illustrate Paceline's service model, Figure 3 contains a pseudo code example of the logic that an adaptive real-time application might use, in this case an adaptive videoconferenc-

ing client. The client calls the `send_video_frame` function to send a video-frame message. This function sends the message with an importance specified using an application-specific utility measure, reflecting the relative importance of individual frames to perceived quality. If congestion control restricts the rate of the stream, the client will cancel messages of low importance when their utility has expired, while messages of high importance will be sent. For messages of equal importance, Paceline breaks the tie according to position. Paceline's service model provides a clean interface for rate adaptation to match application demands with network conditions, instead of committing messages to the network transport (that is, socket buffer) and then suffering from transport queuing delays.

To benefit from Paceline's data-service model, applications have to develop domain-specific adaptation policies. High-definition videoconferencing, for example, has two dimensions for adaptation: spatial and temporal quality. For each ADU, the application calculates an importance value to estimate its contribution to video quality. Each ADU represents a video layer and uses a Paceline message.

In addition to these two quality dimensions, higher-level indicators can be incorporated, such as the active tab,

Figure 3. Adaptive videoconferencing client.

```
send_video_frame (player, stream, frame) {
    /* Set message data and length */
    msg_init.data      = frame.data;
    msg_init.length    = frame.data_len;

    /* Set message importance */
    msg_init.importance = get_importance(frame);
    msg_init.virtual_time = get_virtual_time(frame);
    msg_init.sent       = video_frame_sent;

    /* Sending a frame with cancellation */
    stream.msg_create(msg_init, &frame.msg_handle);
    stream.msg_write(frame.msg_handle);
    frame.expire_event = expire_video_frame;
    add_timer(frame.deadline,
              frame.expire_event);
}

expire_video_frame (frame, stream) {
    stream.msg_cancel(frame.msg_handle);
}

video_frame_sent (player, frame) {
    cancel_timer(player, frame.expire_event);
}
```

mouse clicks, and position of scroll bar to derive the adaptation policies. Similarly, FPS (first-person shooter) games have limited upload bandwidth to send frequent updates to all game players, especially in epic fights with a large number of players concentrated in one area. Games can reduce the bandwidth requirements using criteria such as proximity, recency, and aim, because players are most likely interested in nearby players or those with whom they have recently interacted.

Paceline is implemented as a user-level library and is layered above standard TCP implementations. As depicted in Figure 4, Paceline's architecture consists of two layers: stream and channel. The *stream layer* manages the application-message queue in Paceline and ensures low latency for data with more influence over quality by enabling adaptation between messages in one stream and across streams. This layer consists of two subsystems: the message framing and fragmentation; and the stream fairness. The channel layer handles the TCP low-level sender-side delays. It consists of the latency controller and the connection manager.

Framing and Fragmentation: Which Chunk to Send?

Fragmentation is the first of several techniques that improve transport

latency. Paceline allows application-level messages of arbitrary size. To decouple transmission delay of potentially large application messages from lower-level queuing delays, the data-transfer mechanism supports sender-side fragmentation of application messages into Paceline chunks, and receiver-side reassembly of chunks back into the original application messages. Chunks are bounded to a small size, typically a fraction of TCP's maximum segment size.

Paceline includes application-level message queues. Unlike lower-level queues that operate in FIFO order, Paceline's message queues are based on priority, so that chunks of newly arrived important messages may quickly preempt older less-important ones. Therefore, chunks of messages with high importance are released to the network faster and observe minimal queuing inside Paceline, as well as minimum application-level transmission delay. Cancellation allows the application to abort a low-importance message if its overall transmission delay is too large.

Stream Fairness: From Which Stream?

Paceline messages (and chunks) are part of a full-duplex stream. Each stream in Paceline has a separate priority queue with chunks ready to be sent.

For fair and timely communication across concurrent streams, Paceline supports two notions of fairness: quality and resource fairness. Resource fairness guarantees fair bandwidth across streams, while quality fairness ensures fair application-level quality. Quality is specified in generic terms but derived from the application level; examples are the frames per second in videoconferencing or the updates per second in online games.

While FIFO or round-robin policies are simple ways of multiplexing data of different streams over the underlying channel, timeliness necessitates a better notion of fairness among concurrent streams, especially when bandwidth is limited. Paceline implemented a fair sharing policy among *active* streams that has data inspired by weighted fair queuing. Each stream has a cumulative *virtual time*, an increasing counter quantifying the resources a stream (messages) has used since it was created. Active streams are organized in order of their virtual time, and chunks are sent from the stream with the minimum virtual time. The important factor regulating how streams are multiplexed is how their virtual times are initialized and adjusted.

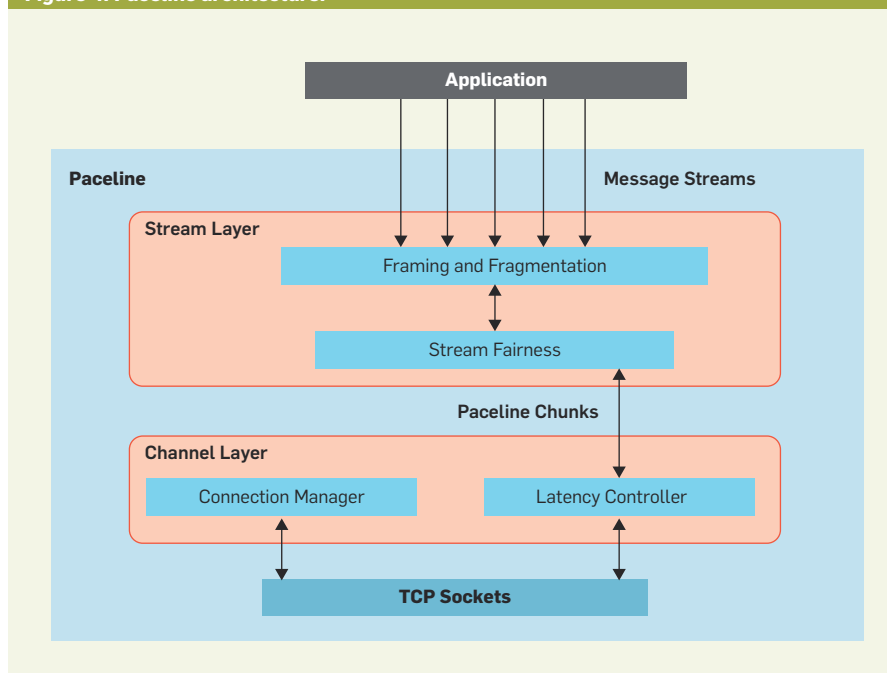
Virtual time initialization is based on two rules:

► **Rule 1 (Fair Start).** When a stream is created its virtual time is set to the minimum virtual time of all active streams to ensure that existing streams do not starve until the new stream catches up in virtual time. If no active streams exist, the virtual time of the newly entered stream is set to the maximum time of all idle streams, or zero if this is the only stream.

► **Rule 2 (Use It Or Lose It).** If a stream becomes active after being idle, the stream's virtual time is set to the maximum of its virtual time and the minimum virtual time of all active streams. This guarantees that no stream, while idle, can save its share of resources for future use.

When a stream transmits a message over the underlying channel, its virtual time is updated according to the virtual times of messages sent, which are based on the application fairness criteria. Conventional resource fairness increments the virtual time based on the size of each message transmit-

Figure 4. Paceline architecture.



ted by the stream. On the other hand, quality fairness increments the virtual time based on an indicator of the application quality of experience, such as the frame rate. By scaling virtual times of streams with different factors, Paceline can allocate different shares to different streams, providing weighted fair sharing.

Latency Controller: How Many Chunks to Send?

To give applications more agility in adapting data delivery, Paceline reduces the amount of committed data in TCP's outgoing buffer and keeps data in its own message queues. The latency controller monitors the progress of the underlying TCP flow and regulates the rate of application data (chunks) delivered to the sender-side TCP. The goal of this controller is to send chunks into TCP fast enough to allow the congestion control to claim the flow's fair share of available bandwidth, but not so fast as to cause an unnecessary amount of FIFO queuing to accumulate in TCP's outgoing socket buffer.

Paceline's controller regulates the writing of application data to TCP in a way that dynamically matches the buffer fill level to a value close to the size of TCP's congestion window ($cwnd$)—namely, $cwnd + 3 \times MSS$. This design implements at the user level the same strategy that was implemented inside the kernel in a previous study and was shown to strike the best balance between latency and throughput.⁸

Paceline has two distinct schemes to estimate the $cwnd$: kernel-assisted and the purely user-level approach. Each has specific advantages. The kernel-assisted scheme, called the PaceK controller, uses information directly from the kernel TCP via the socket API. While this scheme is simple and effective, it requires information that only some implementations of TCP make available. Thus, the PaceK controller is not fully portable. Also, transparent proxies in the network path would likely defeat the PaceK controller's ability to regulate queuing delay, as the TCP socket buffers in the proxies operate independently and can easily become points of major queuing delay if they precede the path bottleneck.

The purely user-level controller in Paceline is called PaceA. Unlike the PaceK controller, it uses only the common TCP socket API available on all major operating systems. Thus, PaceA is more portable (no need to extend or modify kernels), easier to deploy (for example, in relation to firewalls), and avoids problems that result from intermediate proxies. At the user level the value of $cwnd$ is not available so the primary goal of PaceA is to derive $cwnd'$, an estimate of TCP's $cwnd$. PaceA uses application-level acknowledgments (P-ACKs) to measure latency and bandwidth and estimate $cwnd'$ as the $latency \times bandwidth$. More information about the design of PaceA is available elsewhere.⁵

Connection Manager: Which TCP Socket to Use?

In Paceline, the latency controller is the basic technique for limiting TCP latency. In our experiments, however, the distribution of latencies across messages retained a prominent tail, and there was a wide gap (for example, more than a factor of eight) between median and worst-case latencies. We diagnosed the worst-case latencies through a combination of instrumentation in Paceline and packet-trace analysis. Under heavy congestion, TCP can experience back-to-back losses leading to one or more retransmission timeouts. Our diagnosis confirmed that the worst-case latencies were correlated with such episodes of exponential back-off. Similar problems are observed when testing Paceline over wireless links with poor signal strength. To reduce their impact and introduce a ceiling on worst-case performance, Paceline includes a failover mechanism to supplement its basic latency-limiting mechanisms.

Paceline's failover is analogous to the scenario where the user presses the stop/reload buttons in a Web browser upon encountering slow response. Automated failover may sound quite radical, but our evaluation shows that our implementation achieves significant reductions in worst-case latencies while preserving bandwidth fairness. Automated failover resembles removing exponential back-off from TCP, shown to be safe in previous work.

The connection manager maintains a number of back-up TCP sock-

ets and implements failover in a manner that is fully transparent to the application. We switch to a new TCP socket when a threshold is reached. The threshold setting is subject to a trade-off between latency and fairness since replacement channels start in TCP slow-start, possibly resulting in underutilization of the network. The failover threshold is set dynamically using an equation that resembles TCP's RTO (retransmission timeout). A safety margin was added to the failover factor to reduce the number of false positives caused by noise in measurements outside the kernel.

We evaluated Paceline experimentally within a network-emulation testbed. Our measurements are compared against two points of reference: TCP, used to quantify the improvements resulting from Paceline; and SST (Structured Stream Transport),⁶ implemented over UDP with no transport buffer queuing delays. SST provides a rich service model including reliable messaging and congestion control, and it includes the full range of capabilities one might expect from any realistic clean-slate replacement for TCP. Thus, we use SST to approximate a best-case reference point against which to compare Paceline.

In TCP mode, our application still uses the service API of Paceline, but the latency controller is disabled; hence, we send data via TCP as fast as it will allow.

Our network set-up uses the common dumbbell topology, where a set of servers on a LAN connect through a single-bandwidth-delay-constrained link, emulating a congested WAN intra-continental path, to a set of clients on a remote LAN. For the WAN path, we emulate a 30ms RTT delay with a bandwidth limit of 16Mbps or 12Mbps. The WAN bottleneck uses drop-tail queuing with a queue size of twice the bandwidth-delay product of the WAN adding 60ms when the bottleneck becomes congested. The experiments are set up to reflect rather harsh congested conditions, where the bottleneck WAN link is persistently saturated. This is the range in which TCP's performance leaves a lot to be desired.

Transport Performance Summary

Paceline improves upon TCP's weaknesses and maintains its strengths.

Figure 5. Latency threshold versus temporal quality.

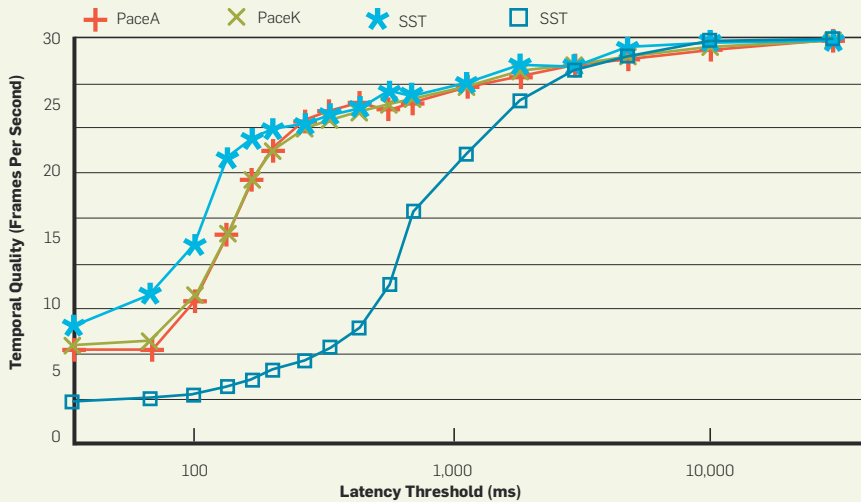
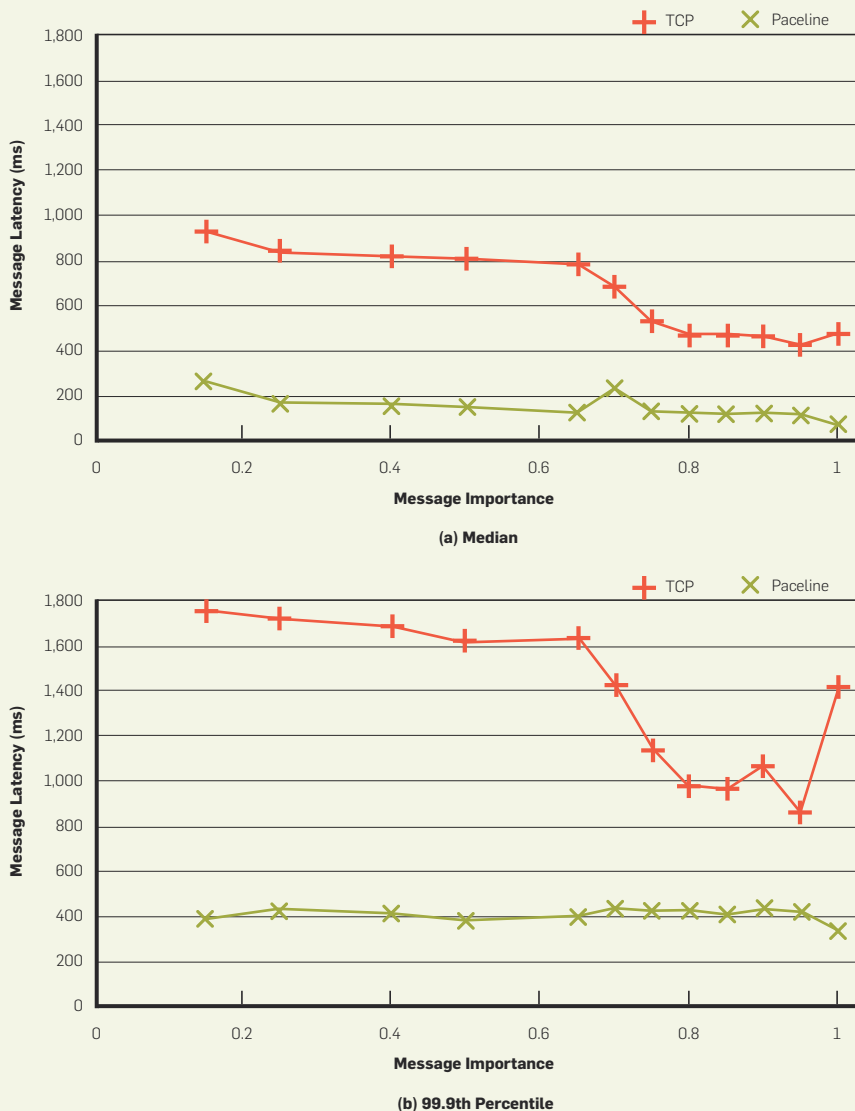


Figure 6. End-to-end message latency based on message importance.



The low-level transport performance compares the transport latency, utilization, and fairness of Paceline with that of TCP. Paceline reduces the median, the 99.9%, and worst-case end-to-end latency by a factor of three to four times. On the other hand, Paceline has similar bandwidth fairness to TCP, high network utilization, and reasonable wire overhead. Paceline is incrementally deployable on the Internet since it shares bandwidth fairly with TCP flows while retaining all latency improvements. (For detailed transport performance, see Erbad et al.⁵)

Application-Level Performance: Does Adaptation Work?

Here, we evaluate the performance of adaptive applications in terms of application-level quality metrics. The evaluation sheds light on the trade-off between average quality and interactivity, and then shows the message latency in Paceline with respect to assigned importance.

Quality and interactivity trade-off. One of the main issues to consider is the nature of the trade-off between overall multimedia quality and interactivity—better interactivity (lower latency) generally comes at the expense of video quality (for example, spatial detail). The following experiments fix the number of flows to eight videos (4Mbps each, extremely congested link) but vary the level of interactivity using a configuration parameter of the *latency threshold* (the amount of time each ADU is given before it expires and gets canceled by the sender) on outgoing messages.

To quantify video performance close to the level of user experience, we measure the frame rate in fps (frames per second). Adaptation in the videoconferencing test application prioritizes ADUs according to two dimensions of video quality: temporal quality (frame rate) and spatial quality, measured with PSNR (peak signal-to-noise ratio) of frames. The video format is scalable so each video frame consists of eight ADUs with one base spatial-layer ADU and seven (progressive) enhancement spatial-layer ADUs. The default adaptation policy is biased toward temporal quality. That is, as the bit rate of a video stream drops, spatial enhancement

ADUs are dropped; and when the spatial quality nears minimum, then further reductions in bit rate will cause dropping of base ADUs, which will result in dropping entire frames (lower temporal quality). Therefore, in the congested settings used for testing, the temporal quality (that is, frame rate) is the quality measure.

Figure 5 shows the average frame rate as the latency threshold is varied. Notice that on the rightmost side of the graph with the highest latency thresholds (tens of seconds), all transports achieve full temporal quality of the video (30fps). The temporal quality when using TCP drops much more rapidly moving leftward (with lower latency thresholds). Even though TCP delivers high throughput, the high transport latency with TCP causes frequent head-of-line delays blocking between low-importance ADUs (spatial enhancements) and high-importance ADUs (base layers). This translates to dropped frames and a much lower fps rate.

The trends exhibited by SST and Paceline are very similar. Recall that SST's implementation completely avoids transport queuing delays. Comparing temporal qualities of Paceline and SST, we see that Paceline also eliminates most TCP sender-side queuing delays. The knees of the Paceline and SST curves in the 100ms–200ms zone indicate that even in this heavily congested network, it is possible for an application such as videoconferencing to keep within the zone of reasonable interactivity with a modest impact on quality. On the other hand, using TCP as the transport results in quality not increasing substantially until well over the 500ms point, which is probably not acceptable for comfortable interaction.

Importance effects on latency. Up to this point, Paceline was shown to be within the zone of responsiveness similar to clean-slate protocols such as SST. This section investigates the effects of importance on message latency. Messages are spread into buckets according to their importance, and the one-way end-to-end latency of the delivered messages is measured in each bucket. Figure 6a presents the median latency, while Figure 6b is the 99.9th percentile latency.

As shown in Figure 6, both TCP (with adaptation) and Paceline have lower median and worst-case latency for important data, with an improvement of more than a factor of two over less-important data. Since TCP commits messages in the kernel send buffer, TCP flows have higher overall latency with a median that is well above the expected latency (275ms). Paceline, on the other hand, keeps the median latency very close to the one-way delay (75ms) for more important data. Paceline also has consistent 99.9th percentile latency due to failover, which is close to 400ms for all messages. The 99.9th percentile latency in TCP is above a second for the majority of messages, reaching almost 1.8 seconds in some cases.

We evaluated quality fairness across streams in Paceline. Video quality is defined by the temporal quality (fps). Figure 7 plots the frame rate of three videos over time. The videos (transferred over three streams) display with identical quality (in terms of frame rates) that changes based on network conditions. It is interesting to note that streams were allocated different bandwidth shares in the same period to achieve equal quality.

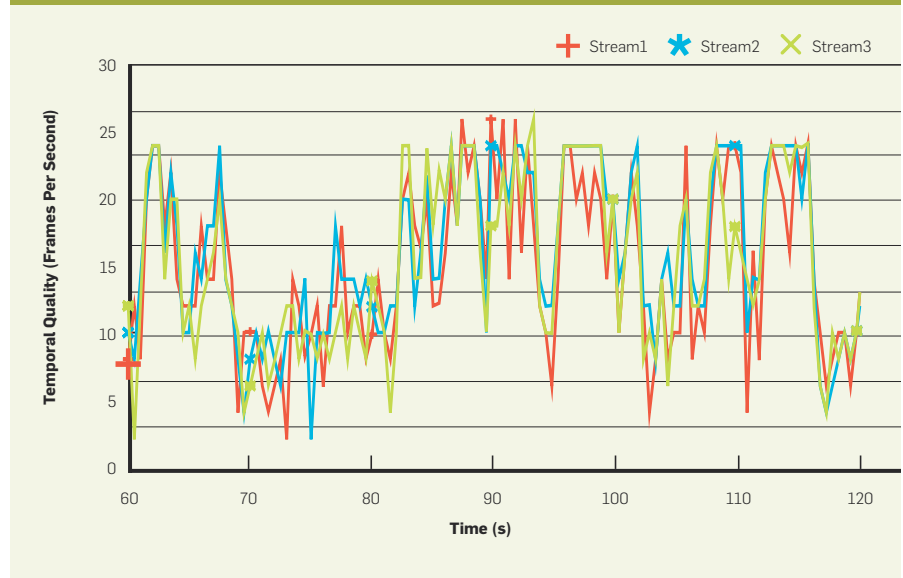
Quality fairness in the Paceline model is completely controlled by application-quality metrics. We provide applications with the notion of importance to control adaptation within streams. Weights and virtual time, on the other hand, specify importance across stream boundaries.

Limitations and Future Work

The Paceline implementation is written in C, and we have been mindful of performance and efficiency from the start. Using QStream, a complete end-to-end implementation of adaptive video streaming, was helpful because the application provided visual and quantitative feedback directly connected to each performance change. Later Paceline was used in research on massive scale gaming, which revealed performance weaknesses not apparent in the video setting. Prominent among these, certain elements of game traffic (state updates) involve very high volumes of small messages, and keeping processing overhead down in this setting is a challenge, particularly in terms of taxing dynamic memory allocators. We have a design to reduce the Paceline memory allocation to at most one per application-level message.

Current transport protocols such as SPDY embrace all-SSL-all-the-time methodology, partly motivated by security but also motivated to mitigate myriad problems caused by middle boxes that are intolerant (intentionally and not) to new protocols. In hindsight, it would have been useful to think of SSL integration from the early stages of Paceline's design. Paceline can perform SSL negotiation at the channel level, amortizing the cost of the initial negotiation. We also need to ensure that encryption happens only when messages are written to the socket in order to avoid canceling encrypted data.

Figure 7. Quality fairness policy: Temporal quality



The Minion architecture of Nowlan et al.¹² has several features that would be beneficial and complementary to Paceline's strengths, such as its SSL strategy and wire-compatible changes to TCP that allow unordered receive.

Paceline reduces sender-side queuing delay by writing the minimum amount to the TCP socket buffer. Recently, the bufferbloat problem has received greatly renewed general attention and interest. The recent algorithm work by Kathleen Nichols and Van Jacobson is a promising step since it is easy to implement and needs no configuration.¹¹ What is still missing, however, is true end-to-end evaluation that quantifies the total combined effect of refinements to TCP, ECN (Explicit Congestion Notification), AQM (Active Queue Management), and adaptive multimedia.

Underlying our interest is what we believe remains an open question: Is there really a line between applications that needs to break from TCP for interactivity reasons? This is a line that has been moving steadily over time.

Lessons Learned

Introducing a new transport layer is challenging because the designer must address several critical issues. First, the transport layer has to improve performance significantly for the target applications in order to justify the extra effort. Second, the enhancements should not negatively affect other traffic types; we adhere to the vision that the Internet should remain a general-purpose infrastructure for a vast array of applications. Finally, the performance improvements at the transport level in terms of latency, fairness, and utilization need to translate to quality improvements at the application level. Paceline has shown positive results in all these aspects.

We tested Paceline extensively using videoconferencing scenarios over WAN settings. Paceline was also used in a small cloud-based game prototype to scale the wide area communication of an Epic-scale game scenario.¹³ Both of these applications show significant improvements in multimedia quality because of the reductions in TCP sender-side delays and the use of adaptation in Paceline.

To ensure we have a general-purpose transport supporting a wide range of applications, we tested using a network

traffic generator simulating different HTTP Web traffic flavors (for example, HTTP1.0 and HTTP1.1, with and without pipelining). The experiment tested Web document downloads while varying the number of objects per page and the page size. Paceline improved the bandwidth utilization using the multi-streaming feature as we increased the number of objects as a result of the automatic pipelining of small transactions on top of the underlying channel.

We evaluated Paceline using transport and application-quality metrics. Transport-level metrics were necessary during the early stages of developing Paceline algorithms, such as the latency rate controller and failover criteria. It was essential at that stage to ensure Paceline did improve latency without reducing fairness and network utilization. As the transport became more mature, however, we needed to ensure these low-level improvements translated into quality improvements for video and other multimedia applications. Verifying application-quality improvements is the limitation of the majority of newly proposed transports.

Finally, effective use of the network required application-level knowledge of quality and importance measures, along with careful tracking of messages to avoid wasting bandwidth. Quality adaptation is an essential transport feature. Paceline enables applications to scale quality with the available bandwidth and to favor data with more influence over quality. Adaptive multimedia applications can provide graceful failure modes with different quality levels instead of rebuffering when bandwidth is limited.

The Paceline implementation is part of the QStream video-streaming system, which is open source and may be downloaded from <http://qstream.org>. More implementation and evaluation details can be found in an earlier article on the topic.⁴

Acknowledgments

Thanks to colleagues at the Network, Systems, and Security lab at the University of British Columbia for their feedback on previous drafts of the article. Special thanks to Mahdi Tayarani Najaran for his significant contributions to the design and implementation of Paceline. Finally, thanks to Terry Coatta

for helping to improve both the style and content of this article. 

Related articles on queue.acm.org

Four Billion Little Brothers?: Privacy, mobile phones, and ubiquitous data collection

Katie Shilton

<http://queue.acm.org/detail.cfm?id=1597790>

VoIP: What is it Good for?

Sudhir R. Ahuja and Robert En

<http://queue.acm.org/detail.cfm?id=1028897>

Data in Flight

Julian Hyde

<http://queue.acm.org/detail.cfm?id=1667562>

References

1. Bharambe, A., Douceur, J. R., Lorch, J.R., Moscibroda, T., Pang, J., Seshan, S. and Zhuang, X. Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. In *Proceedings of the ACM SIGCOMM Conference on Data Communication*, (2008), 389–400.
2. The Chromium Projects. SPDY: An experimental protocol for a faster Web, (2011); <http://www.chromium.org/spdy/spdy-whitepaper>.
3. Erbad, A. Real-time support for interactive multimedia applications (2012); <http://hdl.handle.net/2429/42878>.
4. Erbad, A., Hutchinson, N.C. and Krasic, C. DOHA: Scalable real-time Web applications through adaptive concurrent execution. In *Proceedings of the 21st International Conference on World Wide Web*, (2012), 161–170.
5. Erbad, A., Tayarani Najaran, M. and Krasic, C. Paceline: latency management through adaptive output. In *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*, (2010), 181–192.
6. Ford, B. Structured streams: a new transport abstraction. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, (2007), 361–372.
7. Gettys, J. and Nichols, K. Bufferbloat: Dark buffers in the Internet. *Queue* 9, 11 (2011), 40:40–40:54.
8. Goel, A., Krasic, C. and Walpole, J. Low-latency adaptive streaming over TCP. *ACM Transactions on Multimedia Computing, Communications, and Applications* 4, 3 (2008), 1–20.
9. Krasic, C. A framework for quality-adaptive media streaming: Encode once—stream anywhere. Ph.D. thesis. AAI3119036 (2004).
10. Kuschig, R., Kofler, I. and Hellwagner, H. An evaluation of TCP-based rate-control algorithms for adaptive Internet streaming of H.264/SVC. In *Proceedings of the First Annual ACM SIGMM Conference on Multimedia Systems*, (2010), 157–168.
11. Nichols, K. and Jacobson, V. Controlling queue delay. *Queue* 10, 5 (2012), 20:20–20:34.
12. Nowlan, M., Tiwari, N., Iyengar, J., Amin, S. and Ford, B. Fitting square pegs through round pipes: unordered delivery wire compatible with TCP and TLS. In *Proceedings of the 9th Conference on Networked Systems Design and Implementation*, (2012).
13. Tayarani Najaran, M. and Krasic, C. Scaling online games with adaptive interest management in the cloud. In *Proceedings of the 9th Annual Workshop on Network and Systems Support for Games*, (2010), 9:1–9:6.

Aiman Erbad recently joined the computer science and engineering department at Qatar University as an assistant professor after completing his Ph.D. at the University of British Columbia. His research interests include Web architecture, networking, real-time multimedia, ubiquitous computing, and concurrency support.

Charles “Buck” Krasic works on large-scale processing of video at Google's YouTube. Prior to joining YouTube, he was a professor in the department of computer science at the University of British Columbia. His research led to the development of QStream. Other research interests include multimedia, operating systems, networking, and distributed systems.

© 2012 ACM 0001-0782/12/12



While the ubiquitous SSD shares many features with the hard-disk drive, under the surface they are completely different.

BY MICHAEL CORNWELL

Anatomy of a Solid-State Drive

OVER THE PAST several years, a new type of storage device has entered laptops and data centers, fundamentally changing expectations regarding the power, size, and performance dynamics of storage. The solid-state drive (SSD) is a technology that has been around for more than 30 years but remained too expensive for broad adoption.

That changed with the introduction of consumer products such as the Apple iPad and iPhone, which led to the widespread availability of cheap non-volatile memory. Manufacturers have attempted to use this consumer-grade material to produce SSDs and make them look and act as much as possible like hard-disk drives (HDDs). Under the surface, however, they are completely different.

An HDD uses a head mounted on a mechanical actuator to access rotating magnetic storage media. In contrast, an SSD uses nonvolatile memory (that is, NAND flash) as its storage media. The lack of moving parts and the use of silicon as the media give the device the “solid-state” name. This attribute makes SSDs less fragile than HDDs. As such, SSDs are common today in mo-

bile devices such as smartphones and digital cameras.

Secure Digital (SD) and CompactFlash (CF) memory cards are smaller and less complex versions of an SSD. The key difference between using an SSD and a less complex device is the application’s performance requirements. A digital camera’s need for storage is considerably less demanding than a multicore laptop or server. These differences have significant impact on the design of an SSD.

Both the HDD, the core building block of nonvolatile storage in computer systems today, and the SSD are part of a class of storage called block devices. These devices use logical addressing for accessing data and abstracting the physical media using small fixed contiguous segments of bytes as the

addressable unit. Each block device consists of three major parts: storage media, a controller for managing the media, and a host interface for accessing the media.

NAND Media

Storage media is the key factor behind the performance and cost advantages of SSDs. Most SSDs are built around widely available NAND flash memory, developed in the late 1980s as an electron-based trapped-charge storage media. The NAND cell stores electrons on a capacitor indefinitely in a no-power state. The charge is then sensed by circuitry on the NAND chip.

Writing to NAND flash is accomplished by either adding electrons (programming) or removing electrons (erasing) to the memory cell using high-voltage pulses. NAND flash is read by using a simple analog-to-digital converter as a bias voltage is applied to the cell.

Different types of flash use different numbers of thresholds to determine the value in a cell. Single-level cell (SLC) stores a single bit of data and has two threshold values. Multi-level cell (MLC) stores two bits of data and has four threshold values. Newer flash memories, called Three-level cell (TLC), are able to store three bits of value with eight or more threshold values. In general the value of an “n” bit A/D convertor can be described as: $n = \log_2(\text{\#threshold voltages})$.

A write-verify operation is used to program or erase the NAND flash. A pulse of high voltage is applied to the cell and this process is repeated until

the proper value has been programmed into the cell. The big drawback with this technique is that a cell needs to be erased before it can be reprogrammed. Also, the more levels that a flash cell supports, the smaller and more precise the high-voltage pulses must be. Making these pulses more precise slows the programming of the flash and reduces the write performance.

Disk drives use symmetrical access—the minimum read and write accesses are the same size. NAND flash memory, on the other hand, like most nonvolatile memory, uses asymmetrical access—the size of the minimum read and write to the media is dissimilar. This asymmetrical attribute is a result of the architecture of the memory array. Like most memory, NAND consists of a two-dimensional array with a bit line as one dimension of access and a word line as the other dimension of access. The difference in NAND is that multiple cells share the bit line. This NAND string consists of 32–64 cells. The smallest size of read is typically an 8KB page, based on the word-line length. A write to the array requires linearly programming all the pages along the NAND string, making the smallest write size 32–64 pages. This is called a program/erase block.

A die is a silicon wafer that consists of one-to-four memory arrays of approximately 4,000 blocks and the elements necessary to make the array usable. These non-storage elements consist of: controller logic for managing operations on the memory array; high-voltage generators necessary for

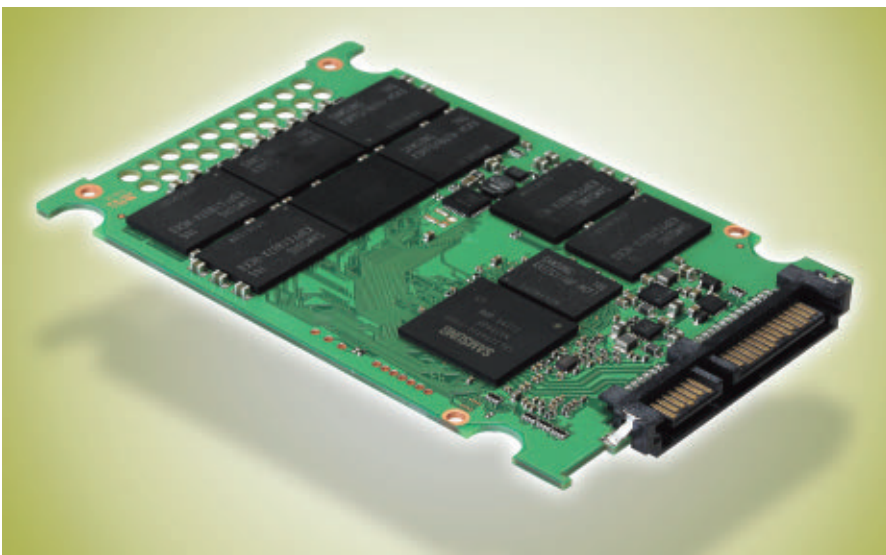
programming and reading from the array; sense detectors for reading the threshold values from the cell; cache buffers for temporarily storing the data bits to and from the memory array; and a high-speed interface for reading and writing data out of the die.

A single die is capable of reading up to 400MB/sec but can write at only 2MB–10MB/sec due to the complexity of programming. Latency is a major benefit of NAND flash. Typical latencies are 20–200 microseconds for reads and 1–10 milliseconds for writes. This performance compares favorably with HDDs, where reads and writes are typically measured in tens of milliseconds.

NAND flash dies have a relatively small capacity, holding up to 16GB per die. An SSD contains 8–256 dies to meet storage requirements for computing. Because multiple dies will be active at the same time, larger-capacity SSDs tend to yield better performance. Multiple active dies can cause potential thermal problems however, as the high-voltage generators in each die are inefficient. During heavy program and erase operations an SSD limits the number of active dies to avoid overheating or excessive peak current draws.

Electron-based storage has many limitations, as is true of all storage media. The biggest limitations are found in the program and erase operations. The high-voltage pulses will burn out the oxide layer, reducing the cells' ability to isolate the electrons. Electrons will also become trapped in the oxide layer, adding resistance to the threshold measurement and causing a misread of the threshold value from the cell.

Burnout is the most misunderstood attribute of NAND flash. Manufacturers often specify the number of recommended program and erase cycles per cell for warranty purposes but do not specify the amount of time that a cell will retain data. Even without excessive use, stored electrons will eventually dissipate from the cell, and the data will be lost. The number of program and erase cycles simply accelerates the time before the data fades. Data can be retained for years in lightly cycled cells, while heavily cycled cells may retain data for only a few months. Long exposure to high temperatures also accelerates the decay of data. Numerous reads of the



same NAND block can cause electrons to escape, changing the value stored within the cell, which leads to a failure mechanism called read disturbance.

As manufacturing technology improves, allowing the cell size to shrink and reducing the cost, the number of electrons stored per cell shrinks, as does the isolation oxide size per cell. This makes it more difficult to meet the performance and longevity of the previous generation. These limitations, as well as many others, make NAND flash an impractical storage technology by itself. Burnout, data fade, and read disturbance are solvable problems for the SSD, however, as constantly and consistently moving data can prevent such failures. Thus, a symbiotic relationship between NAND flash and the controller is necessary for managing and working around the imperfections on the media.

Controller

The flash controller is the critical component that makes imperfect NAND flash robust and reliable. The controller is a complex embedded system with stand-alone processing and firmware for managing all aspects of the SSD. It is designed to protect and control the underlying NAND flash media.

Similar to a disk drive, the flash controller is most commonly implemented as a system-on-a-chip (SoC) design. The controller consists of multiple hardware-accelerated functional blocks, coupled to one or more embedded processor cores that live within a single application-specific integrated circuit (ASIC) die to provide the cheapest controller cost. Large static RAM (SRAM) for executing the SSD firmware is included in the ASIC, but often external dynamic RAM (DRAM) is used for caching both user data and internal SSD metadata. Higher-end SSDs include a backup power system of batteries or capacitors to ensure the user data in the volatile cache is flushed out to the NAND array during unexpected power loss.

Host Interface

The host interface is the physical interface from the host system to the SSD. The majority of SSDs leverage existing storage standards and interfaces, such as serial ATA (SATA) and serial attached SCSI (SAS). They use traditional block-



The flash controller is the critical component that makes imperfect NAND flash robust and reliable.



access protocols. The low-level primitives and high-speed serializer/deserializer are accelerated in hardware, with high-level block-access protocols implemented by firmware running on the embedded processor.

A newer storage interface for SSDs, not used with HDDs, is Peripheral Component Interconnect Express (PCIe). This is the same general-purpose I/O interface used on laptops and servers. This interface is a more efficient interconnect, since it allows systems to phase out the host bus adapter, an additional controller required on SATA and SAS devices to translate these bus protocols. Removing these bus controllers decreases latency and power consumption.

Flash Channel

The flash channel is the controller and interface circuitry dedicated to communication with a physical subset of NAND flash on the SSD. NAND dies are connected to a controller via a parallel I/O interface capable of 400MB/sec of throughput. This interface is shared by four to eight NAND dies, with only one die able to communicate with the flash-channel controller at any one time.

NAND dies with multiple memory arrays are capable of performing multiple operations at once, but they are not fully independent. The die and the channel controller use a lightweight protocol where data is transferred to/from buffers on the NAND die. The controller has to manage and correctly sequence reads, programs, and erase operations to all dies on the channel for the best utilization. Complex array operations, such as program and erase, will render the single memory array or plane busy until the operation is complete. This can take up to tens of milliseconds. Some SSD controllers have dedicated hardware sequencers or microcontrollers with the ability to reorder die operations for optimal performance.

The most common SSD configuration is eight channels, but an SSD controller can have 4–32 channels to meet performance requirements.

Error Correction

NAND flash has higher performance but also a higher bit error rate than other storage media. The higher error rate requires the SSD to correct bit errors

at gigabytes per second, equivalent to the speed of NAND flash. As such, the error-correction hardware responsible for encoding and decoding all reads and writes to the flash is often the largest portion of the SSD controller. Some controllers implement an error-correcting code (ECC) hardware engine for every flash channel to improve parallel performance, while others implement a single ECC engine shared by all the channels to reduce costs.

BCH. The most commonly used ECC is Bose Chaudhuri Hocquenghem (BCH). It is preferred for its speed and ease of implementation, but this comes at the cost of inefficiencies in the redundant storage required for ECC.

Flash-die designers make assumptions on the amount of BCH correctability needed for a given generation of NAND flash and add redundant space within each page and block to account for it. As dies have shrunk with each successive generation of flash, errors and the need for error correction have lifted; the minimum overhead required for correction (rather than storage) has increased five times over the past five years. In addition, controller designs using BCH alone historically assumed that bit failures were uniform. Newer techniques offer more efficiency and correctability based on better understanding of bit failure characteristics and locality of data within the NAND flash die.

LDPC. The use of low-density parity-check (LDPC) code methods along with more advanced information from the NAND die has led to 8–10 times the correctability over previous BCH methods. LDPC has some drawbacks, however: the correction performance is slower; it requires a significant amount of controller die space to implement; and NAND die designers are often reluctant to share information that would make the LDPC work effectively, as the parametric data needed is often considered a trade secret. A controller might include both BCH and LDPC capability and use LDPC only when the BCH techniques fail, ensuring fast performance with high data reliability.

XOR/Scramblers. Another novel error-correction approach taken from HDD arrays is using XOR parity across

NAND flash has higher performance but also a higher bit error rate than other storage media.

a group of NAND dies. This technique should yield better correction with the ability to survive a full NAND die failure. Also, a data scrambler is used to “whiten” the data before it is written to the die. Whitening the data protects against writing certain user data patterns to the NAND die that can cause high bit failures that result from adjacent interference between cells in the memory array.

Controller Firmware

An SSD essentially has a complex file system running internally on the controller. The firmware running this file system is one of the key differentiators among different SSDs today.

Mapping blocks. The primary function of the flash translation layer (FTL) is to map logical blocks from the system to physical NAND pages and blocks. This mapping has the challenge of handling multiple sizes of requests and alignments because of the asymmetrical I/O access limitations of NAND flash. The system uses logical blocks consisting of 512 bytes or 4KB, which then get mapped into 8KB NAND pages, and finally must be written into a block consisting of 64 or 128 pages.

SSDs have no standard technique for overcoming this issue. A common method is statically to map contiguous logical blocks into page-aligned allocation units the size of the NAND page. Once the SSD has enough written allocation units, they are combined into a NAND block-size unit before being written to the flash. Most SSDs today use a derivative of the log-structured file system (LFS) as the basis for the FTL because the append-only write design works well as a result of the erase-before-programming limitation of NAND flash.

Garbage collection. Because writes can be written only to empty blocks, the FTL must maintain a pool of free blocks. If the FTL runs out of free NAND blocks, or if the SSD is inactive for a period of time, the firmware starts performing background garbage-collection operations to reclaim sparsely filled NAND blocks. These NAND blocks are reclaimed by merging the data into new blocks and erasing the old blocks, thus creating a pool of free blocks for use by the FTL.

Early SSDs had performance inconsistency, particularly under excessive stress, because garbage collection either ran out of blocks or used channel/die bandwidth. SSDs today overprovision the physical NAND flash to ensure there are enough free blocks to prevent performance penalties from garbage collection. Most consumer SSDs overprovision less than 5% extra NAND flash, whereas enterprise-class SSDs overprovision up to 50% for performance-critical applications. SSD benchmarks now take into account the impact of garbage collection and require preconditioning the FTL before taking performance measurements.

FTL. The FTL uses a number of methods to optimize performance despite the physical challenges of NAND flash:

- The firmware tracks the number of times each NAND block has been programmed and erased, and it spreads writes evenly across all NAND blocks in the SSD, increasing SSD longevity.

- A certain number of NAND blocks in a die are defective from the die-manufacturing process. Also, blocks can go bad during SSD operation. The FTL must track these bad blocks and substitute good blocks.

- NAND blocks will be marked for garbage collection based on the age of the data in them to avoid data-retention issues.

- The FTL optimizes throughput and die usage. One common method is statically to interleave allocation units across multiple channels to ensure the best possible throughput, as well as the fullest writes of individual blocks.

System Access

Because HDDs have been assumed to be primary storage media, today's software and hardware are engineered to optimize the performance of these storage devices:

- File systems and applications use complicated heuristics to move the mechanical disk head the minimum distance possible, improving reads and writes.

- Adjacent requests are merged into a single larger request in a process called I/O coalescing, in turn building up the large sequential writes that HDDs handle best.

- There is also an assumption that HDDs will use linear logical address-

ing where the beginning of the address range is the outside diameter of the disk platter (the fastest part of the drive), and the end of the address range is the inner diameter (the slowest part of the drive).

These techniques, however, do not optimize SSD usage and can hinder SSD performance:

- In an SSD, the logical addressing ends up being completely random, based on the write-access pattern and how the FTL places the data on the NAND flash. This renders locality-based algorithms ineffective.

- Queuing large numbers of I/O requests to SSDs can lead to inconsistencies in latency and performance (both I/O coalescing and the optimization of disk-head placement require substantial queuing of I/O requests).

File systems and applications need to be rewritten to take true advantage of the performance and benefits SSDs can offer. Linux has done the most work optimizing scheduling for SSDs. The operating system has tunables for turning off I/O coalescing and locality-based scheduling heuristics, thus improving the predictability and performance of SSDs.

Some new interface additions for file systems and applications break the HDD emulation model and assist SSD performance:

- **TRIM/UNMAP.** A common problem with SSDs is that a host file system could erase free data but did not have a way of telling the storage device it no longer needed that data. The TRIM/UNMAP interface lets the SSD clear the logical block addressing (LBA) entries in the FTL, giving it more free space to use in garbage collection and reducing write amplification. OS/X, MS Windows, and Linux have implemented TRIM.

- **Scatter Gather.** Block-access command protocols limit requests to a single contiguous range of logical blocks, requiring excessive command overhead for small random requests serviced by SSDs. Scatter Gather adds commands for "gathering" multiple noncontiguous requests into a single command, reducing the overhead and improving performance.

While solid-state designs clearly benefit from some of the data-management techniques pioneered in the drive industry, clearly, the HDD emula-

tion model will be broken as we move from the world of computer systems designed for rotating mechanical storage to a world of all solid-state storage.

Solid-State Future

In the future, NAND flash memory will inevitably reach physical limitations. NAND dies are continually shrinking to lower costs, creating endurance and reliability issues that cannot be compensated for by the SSD controller or firmware. Newer memory technologies still in their infancy, such as phase-change memory (PCM) and resistive RAM (ReRAM), show great promise in moving beyond such limitations. They do so in part by shedding the erase-before-programming requirement and the asymmetrical access requirement of the NAND flash used in SSDs.

In turn, this progression invariably will continue the evolutionary/revolutionary paradigm seen today in the transition from rotating media to solid-state devices. These new forms of media will no doubt borrow from, and build upon, the techniques implemented in NAND-based SSDs. At the same time, the shift to these newer technologies will also inevitably require moving beyond the techniques developed today to deal with the unique challenges of NAND. New programming models and interfaces will need to be built to take full advantage of new forms of storage media that offer the speed of DRAM coupled with the data retention of flash. 

Related articles on queue.acm.org

Enterprise SSDs

Mark Moshayedi, Patrick Wilkison
<http://queue.acm.org/detail.cfm?id=1413263>

Flash Disk Opportunity for Server Applications

Jim Gray, Bob Fitzgerald
<http://queue.acm.org/detail.cfm?id=1413261>

Flash Storage Today

Adam Leventhal
<http://queue.acm.org/detail.cfm?id=1413262>

Michael Cornwell is director of technology and strategy for Pure Storage. He was previously Sun Microsystems' lead technologist for flash memory and led the creation of the Sun Storage F5100 Flash Array. Prior to Sun, he served as manager of storage engineering for the iPod division of Apple where he was instrumental in the adoption of NAND flash in Apple products, and worked at Quantum Corporation as a storage architect.

© 2012 ACM 0001-0782/12/12

DOI:10.1145/2380656.2380673

Improved performance and a proven deployment strategy make SPDY a potential successor to HTTP.

BY BRYCE THOMAS, RAJA JURDAK, AND IAN ATKINSON

SPDYing Up the Web

TODAY'S WEB BEARS little resemblance to the Web of a decade ago. A Web page today encapsulates tens to hundreds of resources pulled from multiple domains. JavaScript is the technological staple of Web applications, not a tool for frivolous animation. Users access the Web from diverse device form factors, while browsers have improved dramatically. A constant throughout this evolution is the underlying application-layer protocol—HTTP—providing fertile ground for Web growth and evolution but was designed at a time of far less page complexity. Moreover, HTTP is not optimal, with pages taking longer to load. Studies over the past five years suggest even 100 milliseconds

additional delay can have a quantifiably negative effect on Web use,⁹ spurring interest in improving Web performance. One such effort is SPDY, a potential successor to HTTP championed by Google that requires both client and server changes, a formidable hurdle to widespread adoption. However, early client support from major browsers Chrome and Firefox suggests SPDY is a protocol being taken seriously. Though server support for SPDY is growing through projects like mod_spdy¹² truly widespread server adoption is likely to take time. In the interim, SPDY gateways are emerging as a compelling transition strategy, promising to accelerate SPDY adoption by functioning as a translator between SPDY-enabled clients and non-SPDY-enabled servers (see Figure 1). A variety of incentives motivate organizations to deploy and users to adopt SPDY gateways, as described here.

SPDY (pronounced SPeeDY) is an experimental low-latency application-layer protocol²⁷ designed by Google and introduced in 2009 as a drop-in replacement for HTTP on clients and servers. SPDY retains the semantics of HTTP, allowing content to remain unchanged on servers while adding request multiplexing and prioritization, header compression, and server push of resources. Since 2009, SPDY has undergone metamorphosis from press release written and distributed by Google to production protocol implemented by some of the highest-profile players on the Web.

Figure 2 is a timeline of SPDY milestones, first appearing publicly in a

» key insights

- **SPDY seeks to improve Web page load times by making fewer round trips to the server.**
- **Client-side browser support for SPDY has grown rapidly since 2009, and SPDY gateways offer a transition strategy that does not rely on server-side support.**
- **Open SPDY gateways are an opportunity for organizations to capitalize on the behavioral browsing data they produce.**

IMAGE BY VLADITTO



Figure 1. SPDY gateway translates between SPDY-capable clients and conventional HTTP servers.

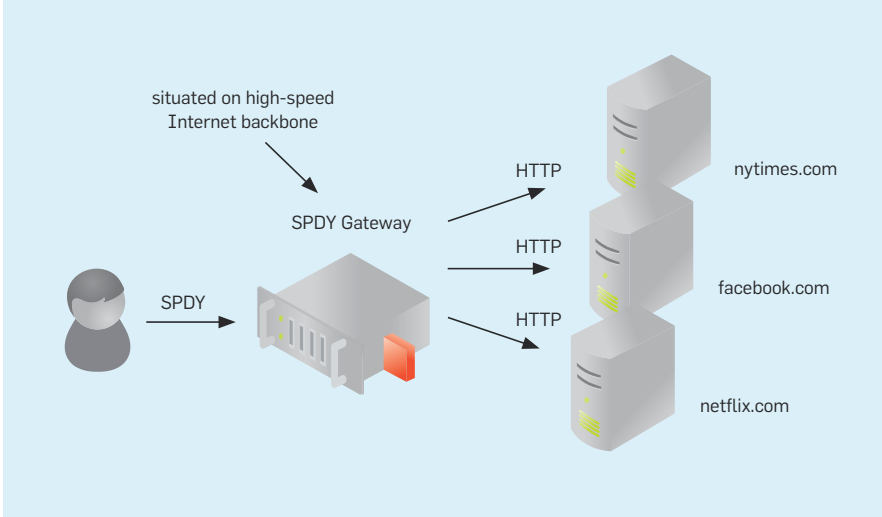
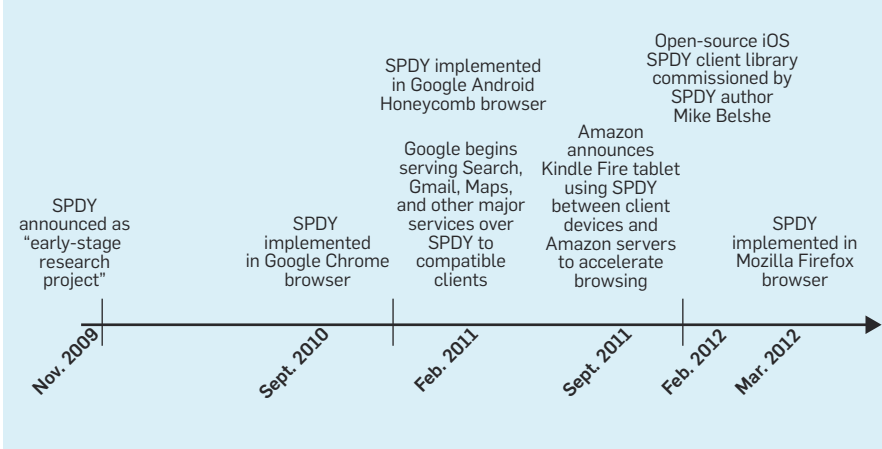


Figure 2. Timeline of SPDY-related development milestones.



November 2009 post⁶ on the Google Chromium blog (<http://blog.chromium.org>) describing the protocol as "an early-stage research project," part of Google's effort to "make the Web faster." By September 2010, SPDY had been implemented in the stable version of the Google Chrome browser,¹⁴ and by February 2011, Google quietly flipped the server-side switch on SPDY, using the protocol to serve major services (such as Gmail, Maps, and search) to Chrome and other SPDY-enabled clients.⁴ In February 2011, the Android Honeycomb mobile Web browser received client-side SPDY support, also with little publicity.¹³ In September 2011, Amazon announced its Kindle Fire tablet, along with the Silk Web browser that speaks SPDY to Amazon cloud servers.³ In January 2012, Mike Belshe, SPDY co-author, commissioned development

of an open-source Apple iOS SPDY client library.²⁰ In March 2012, Firefox 11 implemented the SPDY protocol,¹⁹ which, by June 2012, was enabled by default in Firefox 13,²² bringing combined client-side support (Chrome + Firefox) to approximately 50% of the desktop browser market²⁵ (see Figure 3). SPDY is currently an Internet Engineering Task Force (IETF) Internet draft in its third revision.⁷

SPDY's rapid client adoption is impressive, though server adoption lags considerably. SPDY gateways offer a promising transition strategy, bringing many of SPDY's benefits to clients without requiring server support. The incentive for clients to use SPDY gateways is simple: a faster and more secure browsing experience; SPDY is generally deployed over SSL for reasons discussed later. There are also commercial incentives for companies

worldwide to deploy SPDY gateways on the high-speed Internet. Content-delivery networks have begun offering SPDY gateway services to Web site owners as a means of accelerating the performance (as experienced by users) of their HTTP Web sites.^{1,26} Vendors of mobile devices might deploy SPDY gateways to accelerate the notoriously slow high-latency mobile browsing experience, a marketable feature. Even more intriguing are the incentives for large Web companies to deploy open (publicly available) SPDY gateways to collect and mine rich information about users' Web-browsing behavior, a lucrative commodity in the business of online advertising. Interestingly, the SPDY gateway's ability to aggregate certain critical resources may provide benefits above and beyond regular SPDY, as described later.

SPDY Protocol

SPDY is designed primarily to address performance inhibitors inherent in HTTP, including HTTP's poor support for pipelining and prioritization, the inability to send compressed headers, and lack of resource push capabilities. SPDY's hallmark features—request multiplexing/prioritization, header compression, and server push—are described in the following sections:

Request multiplexing and prioritization. SPDY multiplexes requests and responses over a single TCP connection in independent streams, with request multiplexing inspired by HTTP pipelining while removing several limitations. HTTP pipelining allows multiple HTTP requests to be sent over a TCP connection without waiting for corresponding responses (see Figure 4). Though pipelining has been specified since the 1999 HTTP 1.1 RFC,¹¹ Opera is the only browser that both implements the feature and enables it by default. Other major browsers either do not implement pipelining or disable pipelining by default, as compatibility with older Web proxies and servers is problematic. Besides a lack of widespread adoption, HTTP pipelining also suffers from head-of-line blocking, as the specification mandates resources be returned in the order they are requested, meaning a large resource, or one associated with a time-consuming

back-end process, delays all other resources (see Figure 5).

SPDY implements pipelining without HTTP's head-of-line blocking limitation. Resources transferred through a SPDY connection are carried in annotated "streams," or independent sequences of bidirectional data divided into frames; annotated streams allow SPDY to not only return resources in any order but interleave resources over a single TCP connection⁷ (see Figure 6).

SPDY also includes request prioritization, allowing the client to specify that certain resources be returned with a higher priority than others. Unlike many quality-of-service mechanisms that work on prioritizing packets in queues at lower layers in the network stack, SPDY prioritization works at the application layer, designed to allow the client to specify what is important. One use of prioritization is to request that resources that block progressive page rendering (such as cascading style sheets and JavaScript) be returned with higher priority. Another use of prioritization is to increase the priority of resources being downloaded for the currently visible browser tab while decreasing priority of resources belonging to a currently loading but hidden tab. A further implication of implementing priority at the application layer is that a server can, at least theoretically, prioritize not only the order in which resources are transmitted over the wire but also the order in which resources are generated on the back-end if the task is time intensive.

Header compression. HTTP requests and responses all include a set of HTTP headers that provide additional information about the request or response (see Figure 7). There is significant redundancy in these headers across requests and responses; for example, the "User-Agent" header describing the user's browser (such as Mozilla/5.0 compatible, MSIE 9.0, Windows NT 6.1, WOW64, and Trident/5.0 for Internet Explorer 9) is sent to the server many times over. Likewise, cookie headers, which describe state information about the client, are repeated many times over across requests. Such redundancy means HTTP headers tend to compress relatively effectively. To further improve compression, SPDY seeds an out-of-band compression dic-

tionary based on a priori knowledge of common HTTP headers.²⁹

Server push. This advanced feature of SPDY allows the server to initiate resource transfer, rather than having to wait until a corresponding client request is received; Figure 8 outlines a server using it to push a resource to the client that would be requested imminently regardless. Server push saves superfluous round trips by relying on the server's knowledge of resource dependencies to determine what should be sent to the client.

Performance engineers have employed a range of techniques to try to achieve push-like behavior over HTTP,

though each involves certain shortcomings; for example, data URIs allow inlining resources (such as images) into the main HTML but are not cacheable and increase resource size by approximately 33%. Another technique, Comet, opens a long-held connection to the server⁸ through which arbitrary resources may be pushed but requires an additional initial round trip. Bleeding-edge technologies (such as Web Sockets²⁸ and resource prefetching^{15,21}) also enable push-like behavior but, like Comet, require an additional round trip to establish such behavior. A universal limitation of current techniques is they break resource modu-

Figure 3. Global browser usage share, as recorded by StatCounter (<http://gs.statcounter.com/#browser-ww-monthly-201103-201202>); in February 2012, Chrome had 29.84% and Firefox 24.88%.

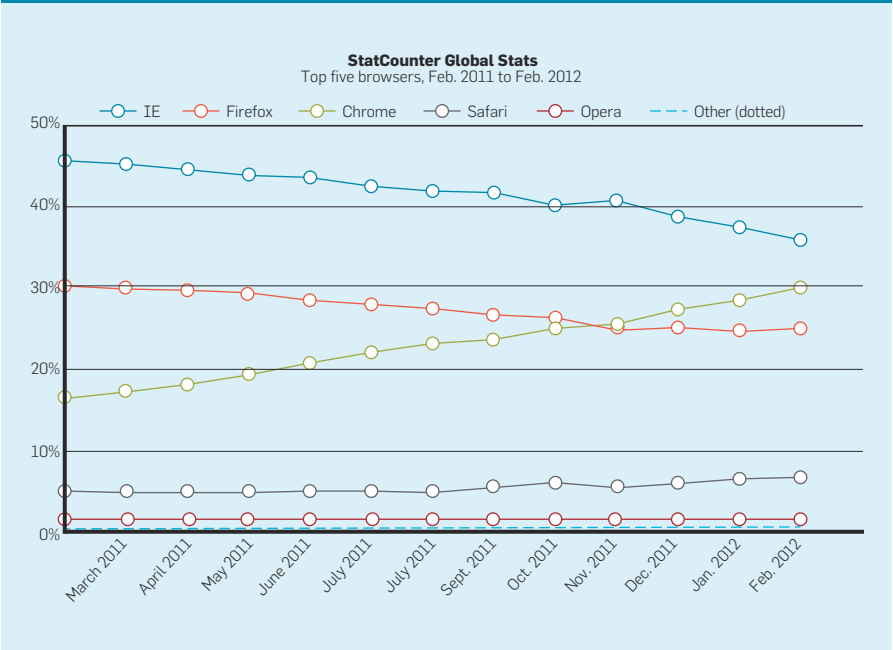
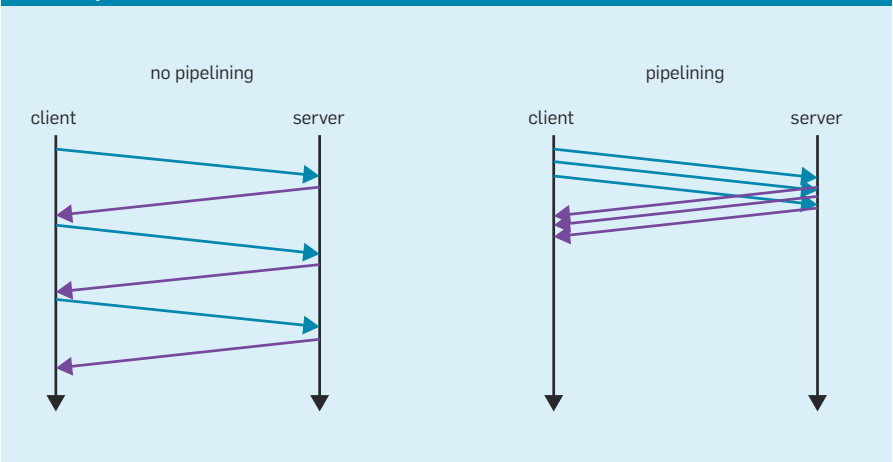


Figure 4. HTTP pipelining allows multiple concurrent requests, reducing the number of round trips to the server.



larity by inserting JavaScript, links, or inlined content into the main HTML document. SPDY push does not require modification of content to support push functionality.

SPDY Security

The SPDY protocol can be run over either a secure (SSL encrypted) or insecure (non-encrypted) transport. In practice, both philosophical views

on the role of Web encryption and pragmatism in handling real-world deployment constraints have led to primarily SSL-encrypted implementations. Mike Belshe, SPDY co-author,⁷ and Patrick McManus, principal SPDY implementer for Firefox, have expressed their interest in seeing the Web transition to a “secure by default” model.^{5,18} Proponents of encrypted SPDY say SSL is no longer computationally expensive¹⁶ and its security benefits outweigh its communication overhead.

The pragmatic reason for deploying SPDY over SSL (port 443) rather than HTTP (port 80) is that transparent HTTP proxies between the client and the server handle HTTP upgrades unreliably.¹³ Transparent HTTP proxies do not modify encrypted traffic on SSL port 443 (as they do on port 80) and so should not interfere with newer protocols like SPDY. The rationale for choosing port 443 over an arbitrary port number is that port 443 appears to traverse firewalls more effectively¹³ (see Figure 9).

SPDY relies on the next-protocol negotiation (NPN)¹⁷ SSL extension to upgrade the SSL connection to the SPDY protocol. NPN is currently an Internet Engineering Task Force Internet Draft and implemented in OpenSSL²³; NPN also allows negotiation of other competing future protocols. SSL implementations that do not currently support NPN simply ignore the request to upgrade the protocol, retaining backward compatibility.

SPDY Performance

Only a handful of (publicly available) studies quantitatively benchmark SPDY against HTTP, all from the same source—Google. Corroborating the following results across different sites represents important future work:

Google Chrome live A/B study. In 2011, Google benchmarked SPDY against HTTP in real-life A/B tests conducted through the Chrome browser. From March 22 to April 5, 2011, Google configured “in the wild” deployments of the Chrome 12 browser to randomly assign 95% of browser instantiations to use SPDY for SPDY-capable sites; the other 5% of instantiations used HTTPS. Google researchers observed a 15.4% improvement in

Figure 5. Head-of-line blocking in HTTP pipelining; a large resource blocks subsequent smaller resources (left), and a slow-to-generate resource blocks subsequent resources (right).

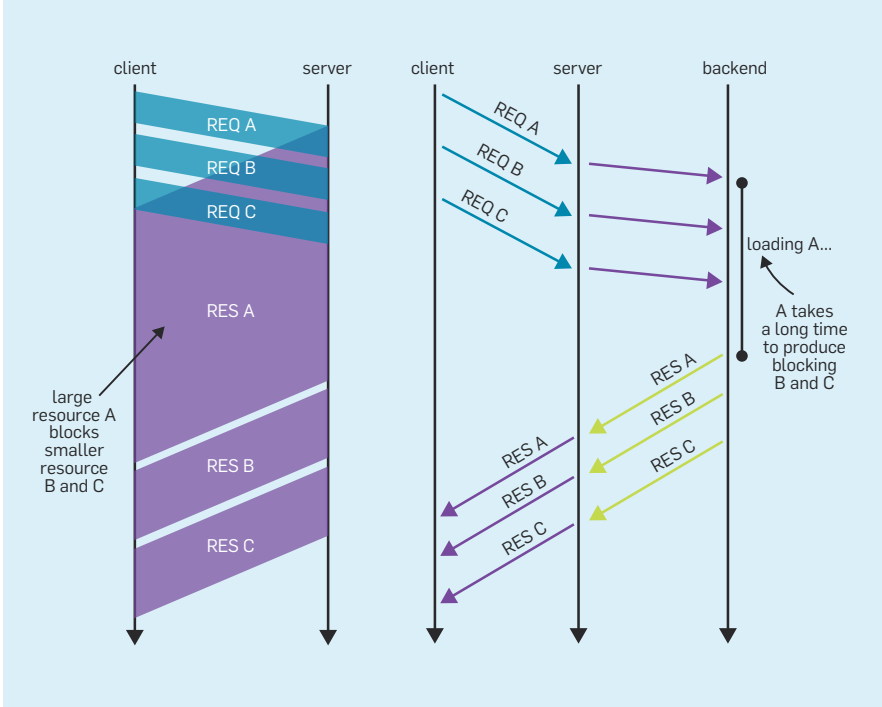
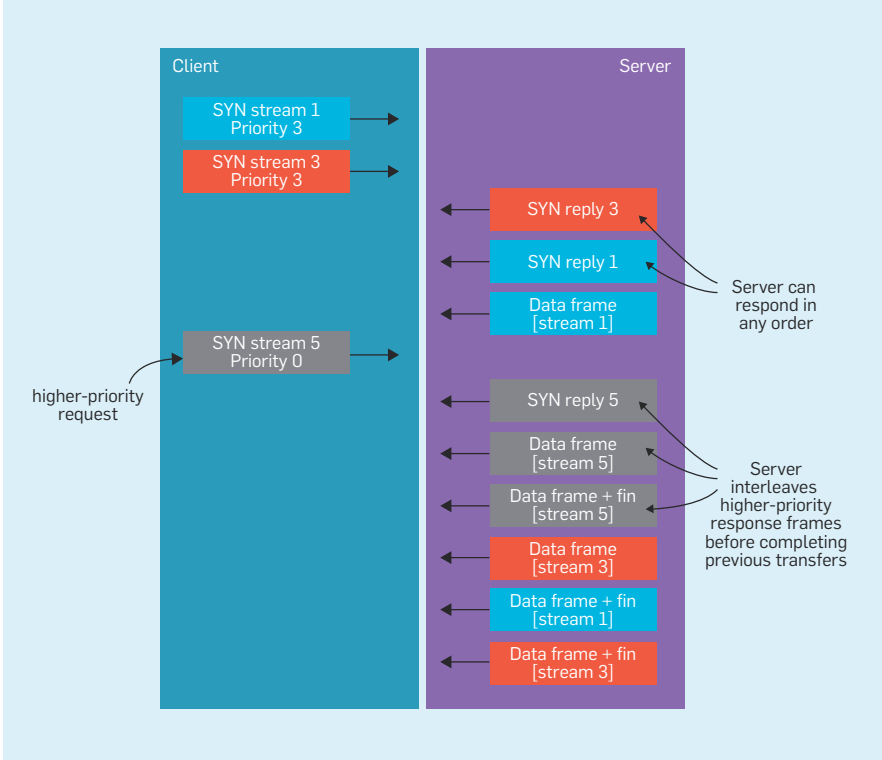


Figure 6. Request multiplexing and prioritization in SPDY; resources are sent in chunks over independent streams that can be interleaved and prioritized.



page-load time across browser instantiations using SPDY,¹³ though a caveat was that domain names in the study were neither recorded nor weighted. Google itself is thought by Google developers to be the current dominant consumer of server-side SPDY technology so is likely overrepresented in these results. Google sites were, in 2011, already heavily optimized, suggesting the stated improvement was likely conservative, though further data is needed for confirmation.

Google's second result from the study was that (encrypted) SPDY is faster than (unencrypted) HTTP for Google's AJAX search; Google researchers provided no further detail.

Google lab tests set one. Google performed a series of laboratory benchmarks of SPDY vs. HTTP under various conditions, though unencrypted SPDY, which would be expected to be faster than encrypted SPDY, was compared against HTTP, despite SPDY deployments being predominantly encrypted.

For simulated downloads of the top 45 pages on the Web (as recorded by Alexa), Google in 2011 reported a 51% reduction in uploaded bytes, 4% reduction in downloaded bytes, and 19% reduction in total packets vs. HTTP.¹³ Uploaded bytes were significantly reduced due to SPDY's header compression and the fact that HTTP headers are amenable to strong compression. Google reported that download bytes were only marginally reduced, as most downloaded content in its tests was not headers and in many cases already compressed. The reduction in total packets is due to both a reduction in overall bytes and the fact that SPDY uses only a single connection, resulting in more "full" packets.

Google lab tests set two. A 2009 Google SPDY white paper¹⁴ described simulated page-load time of SPDY vs. HTTP for the Alexa top 25 Web sites. The first test simulated SPDY vs. HTTP with 1% packet loss over simulated home-network connections. Unencrypted SPDY exhibited 27%–60% improvement in page-load time, and encrypted (SSL) SPDY exhibited 39%–55% improvement in page-load time. A second test determined how packet-loss affected SPDY (unencrypted) vs. HTTP; at 0% packet loss, SPDY was 11% faster, and at 2%

Figure 7. HTTP request and response with example headers; header keys (such as "User-Agent") and header values (such as a particular user agent string) are repeated many times over on a typical connection so make good candidates for compression.

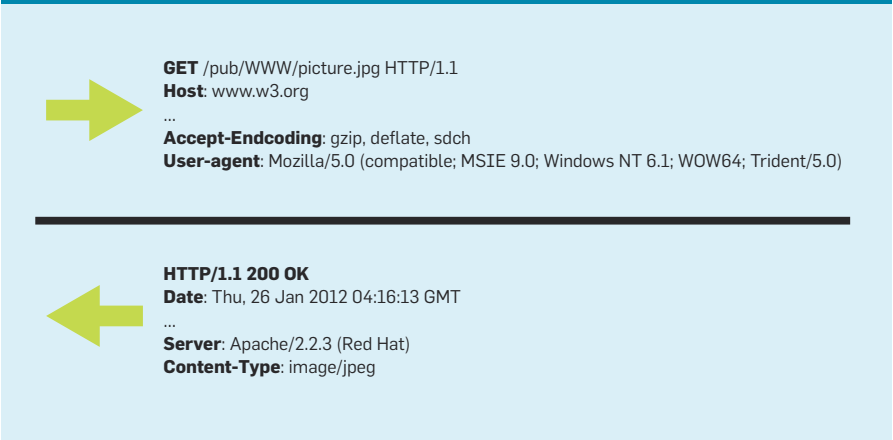
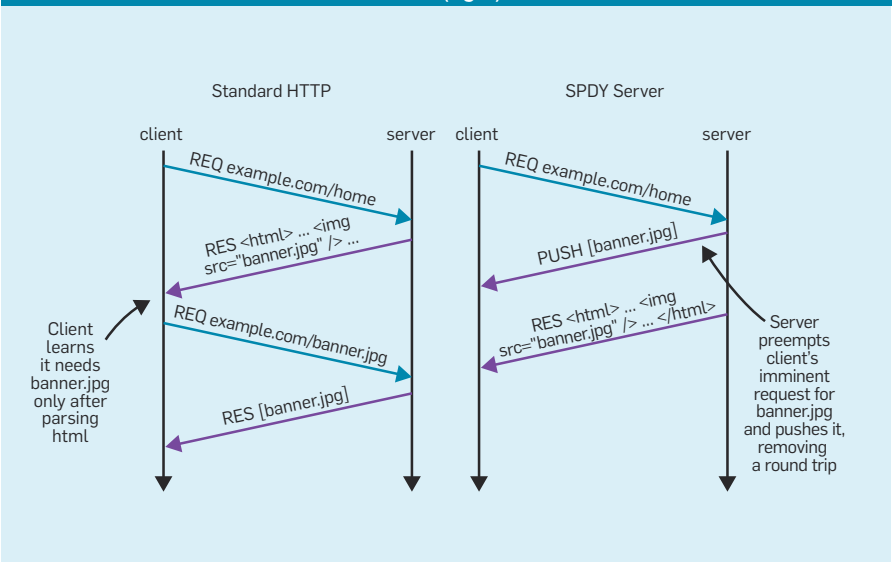


Figure 8. HTTP is unable to push resources to the client, even when it knows the client will require them soon (left); SPDY server push allows the server to initiate the transfer of a resource it believes the client will need soon (right).



packet loss SPDY was up to 47% faster. A third test simulated SPDY vs. HTTP as a function of round-trip time; for example, for a 20ms round trip, SPDY was 12% faster than HTTP, and for a 200ms round trip, SPDY was 26% faster; for a full exposition of these results, see Google.¹⁴

Questions on SPDY performance. There is a conspicuous lack of results describing how SPDY performs on mobile devices. SPDY's dominant performance improvements are due in theory to reduced round trips between client and server. Many mobile-carrier technologies today exhibit latency several times that of their fixed-line and Wi-Fi counterparts. By some projections, the majority of the developing world will have its first Internet experi-

ence through a mobile carrier, proliferating high-latency connections. In theory, SPDY is ideally suited to these high-latency mobile environments, though real-world results are needed for confirmation.

SPDY push-and-request prioritization is also underexplored. For push, work is needed toward determining how aggressive a server should preemptively push resources to the client. For prioritization, no studies exist on SPDY's efficacy in tabbed browser environments where the currently visible tab's downloading resources could be assigned higher priority.

SPDY Effect on TCP Connections

Though SPDY is an application-layer protocol, it involves broader implica-

tions and interesting interactions with the TCP transport layer:

TCP connection proliferation in HTTP. Prior to standardization of HTTP/1.1 in 1999, HTTP/1.0 permitted download-

ing only a single resource over a TCP connection and only four concurrent TCP connections to any given server. HTTP/1.1 introduced persistent connections, allowing connection reuse

for multiple resources. HTTP/1.1 concomitantly reduced maximum concurrent TCP connections from four to two, helping reduce server load and alleviate Internet congestion¹¹ induced by proliferation of short-lived TCP connections at the time. Unfortunately, halving concurrent connections had the adverse effect of reducing download parallelism. HTTP/1.1 envisaged that the newly introduced HTTP pipelining would remedy the problem, but, as described earlier, pipelining proved difficult to implement and suffers from head-of-line blocking, as in Figure 5.

Having only two concurrent connections creates a serious performance bottleneck for modern high-speed Internet connections and complex Web sites. First, TCP slow-start, slowly ramping up usable-connection bandwidth based on number of successfully received packets, is often overly conservative in its initial bandwidth allocation. Several round trips are needed before the connection is saturated, by which time much of the content may have been downloaded already (at a slower-than-necessary rate). Second, a typical modern Web page encapsulates 10s or 100s of resources, only two of which may be requested at any given time. Without HTTP pipelining, requests cannot be queued on the server, so each new request incurs an additional round trip. Because most Web resources are small, the round-trip time to the server often dominates over the time to receive the resource from first to last byte.

Modern browsers break from the HTTP/1.1 standard by allowing six or more concurrent TCP connections to a server. This allocation largely circumvents both previously outlined problems—effective initial bandwidth becoming TCP slow-start constant * 6 (rather than * 2) and fewer round trips incurred due to higher request concurrency. A common practice among large Web properties (such as Facebook, Google, and Twitter) is to “shard”²⁴ a Web page’s resources across multiple domains (such as img1.facebook, img2.facebook, img3.facebook, and img4.facebook) to subvert browser policy and obtain greater concurrency. In a modern browser, a Web page sharded across four domains can receive $4 * 6 = 24$ concurrent TCP connections.

Figure 9. Real-world success rates of upgrading to newer protocols over various port numbers, as measured by Google Chrome’s WebSocket team. Firewalls drop traffic on arbitrary new ports, and transparent proxies inhibit protocol upgrades over port 80.

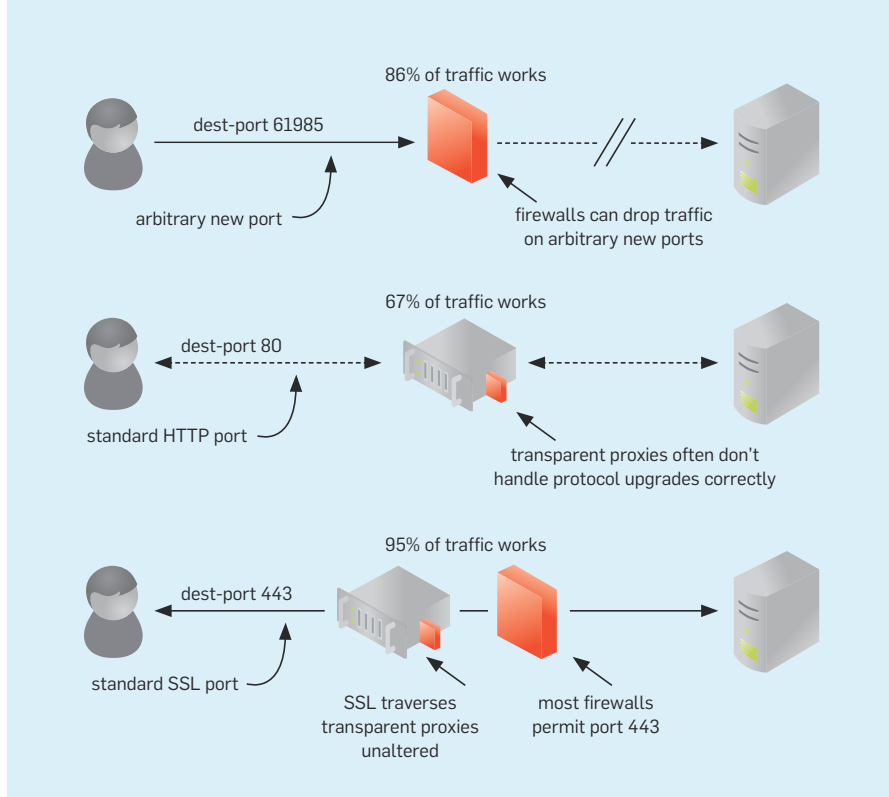
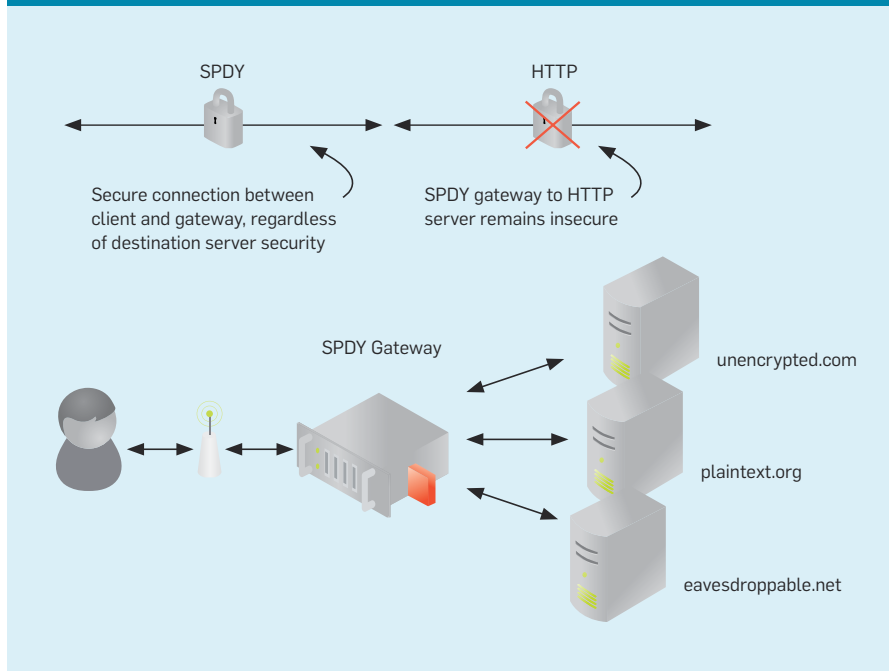


Figure 10. A SPDY gateway offers security between client and gateway regardless of the security of the destination server.



TCP connection proliferation. Increasing concurrent connections through browser policy and sharding can improve page-load time but create other problems. Though highly concurrent connections circumvent an overly conservative slow start on a single TCP connection, they may (in aggregate) exceed total available bandwidth, inducing packet loss. Moreover, the likelihood of losing a critical control packet increases with the number of concurrent connections; for example, the TCP SYN packet, which initiates a TCP connection, has a retransmission timeout measured on the order of seconds if no acknowledgment is received.

Highly concurrent TCP connections also decrease the likelihood of fast retransmit being invoked under packet loss. Fast retransmit is a TCP enhancement that immediately re-sends a packet without waiting for a fixed timeout delay if acknowledgments for several packets subsequent to the lost packet are received. Highly concurrent connections obtain less bandwidth individually than a single long-lived connection and are therefore less likely to receive and acknowledge enough packets in a short enough duration to trigger fast re-

transmit. There is also less “body” in each short-lived connection, increasing the likelihood that any packet loss would occur near the end of a connection where too few acknowledgments exist to trigger fast retransmit.

Finally, highly concurrent TCP connections create more connection states to be maintained at various points in the network, including at network-address-translation boxes, as well as state binding processes to TCP port numbers. In some instances, this state can even cause poorly implemented hardware and software to fail or misidentify the highly concurrent connection openings as a SYN flood (a type of denial-of-service attack).¹⁰

SPDY elephant vs. HTTP mice. The highly concurrent short-lived TCP flows of modern HTTP fall into the category of connections colloquially known as “mice” flows. In contrast, SPDY is able to use a single long-lived “elephant” flow, as it can multiplex and prioritize all requests over a single connection. SPDY therefore retains the benefits of highly concurrent HTTP connections, without detrimental side effects.

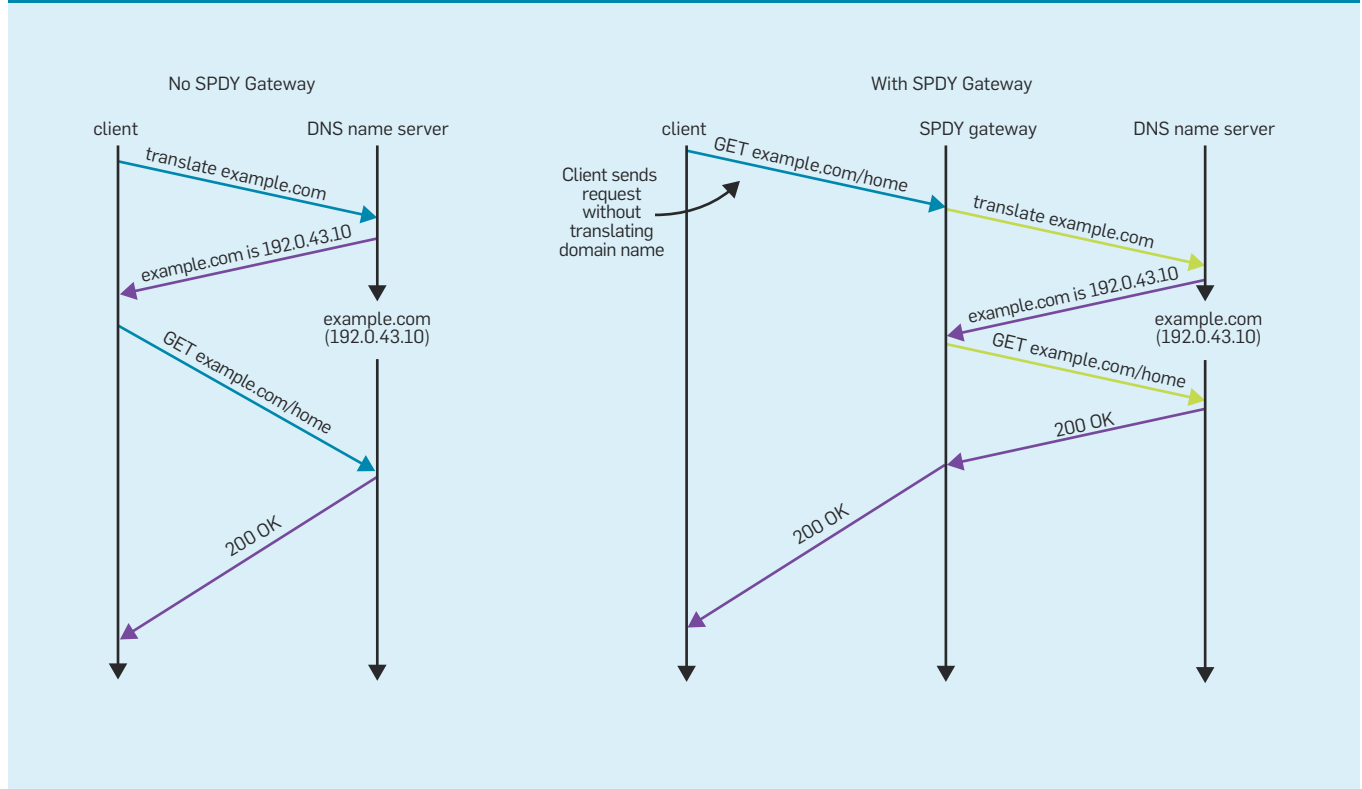
A short-term disadvantage of SPDY’s single-connection approach is inequitable TCP “backoff” compared

to competing applications still using multiple TCP connections; for example, a backoff algorithm that responds to packet loss by reducing the available bandwidth of a connection by 50% will likewise halve the total bandwidth available to an application using a single SPDY TCP connection. The same backoff algorithm applied to an application using 12 concurrent TCP connections would reduce the total available bandwidth to the application by only 4% (1/24) of the connections. Connection proliferation should not be encouraged over the long term, though a short-term mitigation strategy would involve using a small number of concurrent SPDY connections. Long-term research may look to address this issue through smarter backoff algorithms providing equitable treatment to applications, independent of the number of TCP connections.

Transitioning to SPDY

SPDY has been implemented in several popular client browsers, most notably Chrome and Firefox. Though server support for SPDY continues to grow, it has yet to reach the maturity and adoption of client implementations. SPDY gateways are one way to acceler-

Figure 11. A client can delegate DNS lookup to a SPDY gateway, helping minimize round trips.



ate SPDY adoption, providing many SPDY performance and security advantages without requiring SPDY support on the server. A SPDY gateway is an explicit proxy that translates between SPDY-enabled clients and HTTP-only servers. By situating such a gateway on the high-speed Internet, SPDY is used over the slow “last mile” link between the client and the Internet core. The HTTP portion of the connection is in turn isolated to the very-low-latency, very-high-bandwidth link between the gateway and the server, largely mitigating HTTP’s dominant performance inhibitors. In addition to providing a practical, viable SPDY transition solution, SPDY gateways also offer several performance-enhancing features:

Secure connection to gateway, regardless of server-side SSL support. Because SPDY operates over SSL, the client-to-gateway connection is secure, regardless of whether SSL is supported on the destination server (see Figure 10). Though the gateway-to-server connection could remain insecure, clients are protected from common attacks (such as eavesdropping on insecure Wi-Fi access points).

Single client-side connection across all domains. As described earlier, SPDY request multiplexing results in dramatically fewer TCP connections than HTTP browsers in use today. However, clients still require at least one new TCP connection for each new server they contact. A SPDY gateway can achieve even greater efficiency than regular SPDY by multiplexing all of a client’s requests to the gateway over a single TCP connection covering all servers.

A SPDY gateway might still create multiple connections to a given HTTP server to emulate pipelining and avoid head-of-line blocking but isolate these connections to the high-speed/low-latency Internet core. A SPDY gateway may also retain a small pool of TCP connections to popular servers, allowing new client requests to be forwarded immediately without incurring a new TCP connection handshake or slow-start “warm-up.” Likewise, the client needs to perform a single TCP connection handshake only with the gateway and go through the slow-start warm-up only once (as opposed to every time a new server is contacted).



Aside from offering faster browsing as a selling point, Amazon and other potential vendors are likely interested in the data mining and advertising opportunities that come with controlling the gateway.



Delegated DNS lookup. This performance enhancement specific to SPDY gateways entails the gateway performing DNS translations from domain names to server IP addresses on behalf of the client, allowing the client to immediately send a request for a resource to the gateway without knowing the IP address of the server on which it is hosted (see Figure 11). Being situated on the high-speed Internet, the gateway is better positioned to quickly translate the domain name to an IP address; moreover, a gateway that serves a large number of users can cache the IP addresses of popular domains.

Intelligent push. A SPDY gateway can exploit its large user base to infer resource dependencies, even across domains. A regular SPDY-capable server has a limited view of a user’s browsing behavior, isolated to the server itself. A gateway sees a user’s requests for all servers so it can infer complex patterns of cross-domain navigation; for example, the gateway could determine that 95% of users issuing a Google search query for “Twitter” proceed to twitter.com, and, given this knowledge, the gateway then preemptively pushes resources from the twitter.com homepage to the user. In 2011, Amazon reported the Silk browser on the Kindle Fire tablet already performed intelligent push of this nature, called by Amazon “predictive rendering.”²

Caching. Like a transparent proxy, a SPDY gateway can cache resources such that subsequent requests for the same resource are served without contacting the origin server.

SPDY gateways, a permanent fixture? This description of SPDY gateways highlights that in some respects gateways offer more attractive features than SPDY directly between clients and servers, including four notable functions: further reduction in TCP connections over the last mile; pre-warming of TCP connections; delegation of DNS translations to the fast Internet core; and intelligent push and resource caching. We suggest that gateways may have a persistent role on the Web, beyond mere transition strategy.

Future SPDY Gateways

Several companies have deployed large SPDY gateways. Perhaps most

notable is the gateway used by the default Silk browser on the Amazon Kindle Fire tablet²; Silk proxies much of a user's Web traffic through an Amazon SPDY gateway deployed on the Amazon Web Services cloud infrastructure. Other examples are content-delivery-network/Web-acceleration providers Contendo¹ and Strangeloop,²⁶ both offering SPDY gateways as a service to HTTP content providers.

Device-specific SPDY gateways. Amazon's decision to couple the Kindle Fire Silk browser to its own proprietary SPDY-based gateway begs the question: Could, and will, other major providers do the same? Could there be, say, an Apple SPDY gateway for iPhones and iPads or a Google SPDY gateway for Android devices in the future? Could such gateways be in the works already? The potential performance advantage of SPDY gateways is particularly intriguing on such resource-constrained mobile devices. The controlled "appliancized" nature of the devices and their operating systems would also simplify vendor implementation. Aside from offering faster browsing as a selling point, Amazon and other potential vendors are likely interested in the data mining and advertising opportunities that come with controlling the gateway.

Open SPDY gateways. Beyond device-specific gateways lies uncharted though potentially lucrative territory—open SPDY gateways—that, like an open proxy, are usable by anyone, independent of device or platform. Major Web companies have demonstrated that free and universal services can be made profitable through related targeted advertising opportunities. So, could SPDY gateways be turned into another free, universal service rendered profitable through better-targeted advertising?

A limitation Web advertisers face today is a restricted view of user activity on domains beyond their direct control. A SPDY gateway provides a vantage point from which to observe all of a user's Web activity, not just on domains under the advertiser's control. Major Web companies like Facebook and Google track users across the Web on third-party sites through partner advertising scripts and other embeddable features (such as the "Like" but-

ton), but the picture is incomplete. An open SPDY gateway would provide advertisers missing pieces from the browsing-behavior puzzle that could be fed back into targeted-advertising algorithms. While much the same could be done using device-specific SPDY gateways, an open SPDY gateway would provide insight into a much larger user population. Interesting to consider therefore is whether SPDY gateways (much like search) could become a universal service accessible through a broad range of devices.

Conclusion

SPDY is a high-performance application-layer protocol and potential successor to HTTP. Clients have been quick to adopt it, though server implementations lag. SPDY gateways are helping accelerate SPDY adoption by removing the need for SPDY support on the server. A range of compelling incentives exists for deploying SPDY gateways that are only beginning to be explored. Beyond just a transition strategy, SPDY gateways have performance characteristics that make them attractive for longer-term use. Whether such long-term advantages compared to SPDY support on the server are substantial enough to warrant retaining SPDY gateways is an open question.

Acknowledgments

This work is supported in part by an Australian Government Australian Postgraduate Awards scholarship and Commonwealth Scientific and Industrial Research Organisation Office of the Chief Executive scholarship. The authors would also like to thank the anonymous reviewers for their valuable comments and suggestions. ■

References

1. Akamai. *Akamai Acquires Contendo*. Press Release, Mar. 2012; <http://www.akamai.com/cotendo>
2. Amazon. Amazon Silk FAQs; <http://www.amazon.com/gp/help/customer/display.html/?nodeId=200775440>
3. Amazon. Introducing Amazon Silk; <http://amazonsilk.wordpress.com/2011/09/28/introducing-amazon-silk>
4. Belshe, M. SPDY on Google servers? Jan. 2011; [https://groups.google.com/forum/?fromgroups#searchin/spdy-dev/SPDYS20on\\$20Google\\$20servers?\\$20/spdy-dev/TCOW7Lw2scQ/INuev2A-ixAJ](https://groups.google.com/forum/?fromgroups#searchin/spdy-dev/SPDYS20on$20Google$20servers?$20/spdy-dev/TCOW7Lw2scQ/INuev2A-ixAJ)
5. Belshe, M. SSL: It's a matter of life and death. Mike's Lookout blog, May 28, 2011; <http://www.belshe.com/2011/05/28/ssl-its-a-matter-of-life-and-death/>
6. Belshe, M. and Peon, R. A 2x faster Web. The Chromium Blog, Nov. 11, 2009; <http://blog.chromium.org/2009/11/2x-faster-web.html>
7. Belshe, M. and Peon, R. SPDY Protocol. Chromium

- Projects, Feb. 2012; <http://dev.chromium.org/spdy/spdy-protocol/spdy-protocol-draft3>
8. Bozdog, E., Mesbah, A., and van Duersen, A. A comparison of push and pull techniques for AJAX in Web site evolution. In *Proceedings of the Ninth IEEE International Workshop* (Paris, Oct. 5–6). IEEE Computer Society, Washington, D.C., 2007, 15–22.
9. Brutlag, J. *Speed Matters for Google Web Search*. Technical Report, 2009; http://services.google.com/fh/files/blogs/google_delayexp.pdf
10. Eddy, W. *TCP SYN Flooding Attacks and Common Mitigations*. Internet Engineering Task Force, Aug. 2007; <http://tools.ietf.org/html/rfc4987>
11. Fielding, R. et al. *Hypertext Transfer Protocol—HTTP/1.1: Connections*. World Wide Web Consortium, June 1999; <http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html>
12. Google Inc. mod-spdy: Apache SPDY module. May 2012; <http://code.google.com/p/mod-spdy/>
13. Google Inc. SPDY essentials. Dec. 2011; http://www.youtube.com/watch?feature=player_detailpage&v=TNBkxA313kk#t=2179s
14. Google Inc. The Chromium Projects. *SPDY: An Experimental Protocol for a Faster Web*. White Paper, 2009; <http://www.chromium.org/spdy/spdy-whitepaper>
15. Komoroske, A. Prerendering in Chrome. The Chromium Blog, June 2011; <http://blog.chromium.org/2011/06/prerendering-in-chrome.html>
16. Langley, A. Overclocking SSL. Imperial Violet Blog, June 25, 2010; <http://www.imperialviolet.org/2010/06/25/overclocking-ssl.html>
17. Langley, A. *Transport Layer Security Next Protocol Negotiation Extension*. Internet Engineering Task Force, Mar. 30, 2011; <http://tools.ietf.org/html/draft-agl-tls-nextprotoneg-02>
18. McManus, P. Maturing Web transport protocols with SPDY and friends. Video of SPDY Talk at Codebits, eu, Nov. 2011; <http://bitsup.blogspot.com.au/2011/11/video-of-spdy-talk-at-codebitseu.html>
19. McManus, P. SPDY brings responsive and scalable transport to Firefox 11. Mozilla Hacks blog, Feb. 2012; <http://hacks.mozilla.org/2012/02/spdy-brings-responsive-and-scalable-transport-to-firefox-11/>
20. Morrison, J. SPDY for iPhone. GitHub, Inc., Jan. 2012; <https://github.com/sorced-jim/SPDY-for-iPhone>
21. Mozilla Developer Network. Link prefetching FAQ. Mar. 2003; https://developer.mozilla.org/en/Link_prefetching_FAQ
22. Nyman, R. Firefox Aurora 13 is out—SPDY on by default and a list of other improvements. Mar. 19, 2012; <http://hacks.mozilla.org/2012/03/firefox-aurora-13-is-out-spdy-on-by-default-and-a-list-of-other-improvements/>
23. OpenSSL. OpenSSL Cryptography and SSL/TLS Toolkit. Mar. 2012; <http://www.openssl.org/news/changelog.html>
24. Souders, S. Sharding dominant domains. Steve Souders blog, May 12, 2009; <http://www.stevesouders.com/blog/2009/05/12/sharding-dominant-domains/>
25. StatCounter. StatCounter GlobalStats, Feb. 2012; <http://gs.statcounter.com/#browser-ww-monthly-201102-201202>
26. Strangeloop Networks. Strangeloop. Mar. 2012; <http://www.strangeloopnetworks.com/products/overview/>
27. The Chromium Projects. SPDY, Mar. 2012; <http://dev.chromium.org/spdy>
28. World Wide Web Consortium. *The WebSocket API: Editor's Draft 29 August 2012*; <http://dev.w3.org/html5/websockets/>
29. Yang, F., Amer, P., Leighton, J., and Belshe, M. *A Methodology to Derive SPDY's Initial Dictionary for Zlib Compression*. University of Delaware, Newark, DE, 2012; <http://www.eecis.udel.edu/~amer/PEL/poc/pdf/SPDY-Fan.pdf>

Bryce Thomas (bryce.m.thomas@gmail.com) is a Ph.D. candidate in the Discipline of Information Technology at James Cook University, Townsville, Queensland, Australia.

Raja Jurdak (raja.jurdak@csiro.au) is a researcher in the Commonwealth Scientific and Industrial Research Organisation and a professor in the University of Queensland, Brisbane, Australia.

Ian Atkinson (ian.atkinson@jcu.edu.au) is a professor and director of the eResearch Centre of James Cook University, Townsville, Queensland, Australia.

DOI:10.1145/2380656.2380674

Computers interacting with, not imitating, humans is the way forward.

BY ROBERT M. FRENCH

Moving Beyond the Turing Test

ALAN TURING WOULD be 100 years old this year. In 1950 he wrote a seminal paper in which he proposed an operational definition of machine intelligence designed to sidestep the philosophical quagmire of what it means to think.¹⁹ Turing proposed pitting a computer against a human in an “imitation game.” The computer and human are placed in separate rooms and connected by teletype to an external interrogator who can ask any imaginable question of either entity. The computer tries to fool the interrogator into believing it is the human; the human tries to convince the interrogator he or she is the human. If the interrogator cannot distinguish the computer from the person, the computer is judged to be intelligent. This simple test has come to be called the Turing Test.

In the early years of research on artificial intelligence, the test was taken very seriously,^{2,8} especially because many researchers believed truly intelligent machines were just around the corner.^{13,17} But as the 1950s to the 1980s came and went and machines were still no closer to passing the Turing Test, AI researchers began to

realize how difficult the problem of simulating human cognition would actually be.¹² It became clear that human cognition emerges from a complex, tangled web of explicit, knowledge-based processes and automatic, intuitive “subcognitive” processes,¹⁰ the latter deriving largely from humans’ direct interaction with the world. Presumably, by tapping into this subcognitive substrate—something a disembodied computer did not have—a clever interrogator could unfailingly distinguish a computer from a person.^{5,6} The hope faded that machines would soon be in a position to pass such a test,⁷ and serious researchers in AI focused their energy elsewhere.^{9,20}

In the past decade, however, significant innovation in computer technology and data capture have brought the Turing Test back into focus. That technology, along with vast information resources that became available at the same time, have potentially brought computers closer than ever to passing the test. But, in spite of these developments, the Turing Test still presents significant hurdles, including some unrelated to machine intelligence. Questions based on largely irrelevant aspects of humans’ physiognomy, quirks in their visual, auditory, or tactile systems, and time required to complete various cognitive tasks can be devised to trip up a computer that has not lived life as we humans have with bodies like our own.

In what follows, I argue we need to put aside the attempt to build a

» key insights

- The time has come to bid farewell to the Turing Test.
- Attempting to build a machine to pass a no-holds-barred Turing Test is not the way forward in AI, regardless of recent advances in computing technology.
- Building machines that would never pass a Turing Test, but that can interact with humans in a highly meaningful, informative way, is the way forward in AI.

machine that can flawlessly imitate humans; for example, do we really need to build computers that make spelling mistakes or occasionally add numbers incorrectly, as in Turing's original article,¹⁹ in order to fool people into thinking they are human? So, rather than require a machine to pass a Turing Test and try to proscribe questions that are unfair or inappropriate to judging its intelligence, we should accept the computer as a valid interlocutor and interact with it as an interactive, high-level, sophisticated information source.

Big Data

If we set aside the attempt to build a machine that can pass the Turing Test, can we still make progress in AI that is, nonetheless, in the spirit of the Turing Test? Let us start by considering two fundamental changes that have occurred in recent years—the availability of vast quantities of data of all sorts and the increased speed and power of machines to analyze that data.

The amount of data now available to machines would have been simply unimaginable as recently as 15 years ago; for example, a full visual and auditory recording of 85% of the waking life of an infant from birth to age three exists.^{15,16} Researchers are also developing (and wearing) sophisticated “life-experience” recording devices that allow individuals to record all of the visual and auditory (and potentially, olfactory and tactile) information they experience throughout the day.¹ This means all of the words you might ever utter, hear, read, or write could be stored somewhere as data. Moreover, it could be multiplied by the thousands or even millions of other individuals who also choose to record their lives. To this, add all other available sources of information, from Twitter feeds to Wikipedia, from Facebook to blogs on every conceivable subject, and much, much more.¹⁸ Equally important is the explosion of new algorithms to retrieve, analyze, correlate, and cross-reference this sea of data.

It is reasonable to assume that all of it, appropriately analyzed, would allow a computer to answer questions, including those about people's

reactions to events, as well as their emotions and feelings, it would have had no hope of answering appropriately a decade ago. But there will forever remain “unfair” questions on the Turing Test, such as one from a recent commentary⁴: “Hold up both hands and spread your fingers apart. Now put your palms together and fold your two middle fingers down till the knuckles on both fingers touch each other. While holding this position, one after the other, open and close each pair of opposing fingers by an inch or so. Notice anything?”

Try it yourself. Simply by doing the experiment, you will discover the fact (completely irrelevant as regards intelligence) that you cannot separate your two ring fingers. But how would a computer without a body ever answer the question or a million others like it? And even if, by trolling the Web, someone had reported the answer to this particular body-dependent experiment, there are thousands of other quirky facts, some related to cognitive abilities (such as computation time for multiplication of multi-digit numbers and misspelled words), some with absolutely nothing to do with intelligence, that would trip up a computer. Attempting to define which of these questions is fair or unfair for a Turing Test is not only contrary to the spirit of the Test as originally proposed by Turing but also an endeavor necessarily doomed to failure. My view is: Don't try; accept that machines will not be able to answer them and move on. The point is essentially the same one I made in an earlier essay,⁵ that it would be “essentially impossible for a machine that has not experienced the world as we have to pass the Turing Test.” This observation in no way implies renouncing the goal of building intelligent machines. It suggests merely that we renounce the Turing-inspired goal of building intelligent machines that mimic our own behavior so perfectly that we would not be able to distinguish them from ourselves.

Evolution of Brute Force

The human brain relies on 10^{11} neurons, each with 10^3 synapses, all working in concert to produce cognition. At the lowest level, the brain is

-- -- -- -- --
-- -- -- -- --
-- -- -- -- --
-- -- -- -- --
-- -- -- -- --

> It seems you
are taking
a certain tone
with me.
An agitated,
human tone...<

>> Really? You
are somewhat
reticent
yourself.<<

> I never said
you were
reticent.<

>> I know.<<

> Smug.<

>> Moi*?<<

> I think
you might be
human.■

-- -- -- -- --
-- -- -- -- --
-- -- -- -- --
-- -- -- -- --
-- -- -- -- --

```

- - - - -
- - - - -
- - - - -
- - - - -
- - - - -

```

> Are you
extremely bored
answering
the same
questions?<

>> For all you
know, I'm
optimized for
repetitive
tasks! But
in all
seriousness,
it's kinda
fun.<<

> If I were
you I'd be
pretending very
hard to be a
computer.<

>> please restate
the question.
=/ <<

> haha lol etc.■

```

- - - - -
- - - - -
- - - - -
- - - - -
- - - - -

```

indisputably performing “mindless” brute-force calculations. In 1997, IBM’s Deep Blue, rated at the time as one of the 300 fastest supercomputers in the world, beat Gary Kasparov, the player generally considered the greatest chess player in history. Deep Blue’s operation was a quintessential example of brute-force search, evaluating some 200 million board positions each second. So, what exactly is the difference between the brute-force computation done by humans and the brute-force computation done by machines? This is a very tricky issue, and there is certainly no simple answer. However, part of the answer involves how brute-force computation evolves.

Imagine that, for some reason, playing high-quality computer chess was essential to human survival. Brute-force search, as practiced by Deep Blue, would likely evolve, by means of automatic techniques akin to genetic algorithms,^{11,14} as well as by explicit human development of ever-more-powerful computer-chess-playing heuristics. People often overlook the fact that Deep Blue’s search of hundreds of millions of board positions per second is inefficient in the extreme, since almost all the positions it considers are completely uninteresting and, therefore, examining them at all is a complete waste of its resources. Consequently, in an evolutionary struggle for survival, such “mindless” brute-force searching would quickly lose out to techniques that channeled brute-force search in ever-more-efficient ways. This is exactly what has happened. Today, there are programs with Elo ratings higher than any human chess player ever and that run on handheld computers. One of the most powerful, Pocket Fritz 4, evaluates “only” 20,000 board positions per second, some four orders of magnitude less than Deep Blue (http://en.wikipedia.org/wiki/Pocket_Fritz).

It is not implausible to imagine this kind of evolution could lead to the emergence in computers of internal representations of board positions and ever-better ways to process these representations. As the internal representations become more complex, better organized in relation to

each other, and processed in ever more sophisticated ways, is it so unreasonable to imagine the gradual emergence of the kind of complexity that would justify the label of a minimal understanding of certain board positions? The bedrock of all understanding is, after all, the ability to construct, contextualize, and make use of internal representations of data.

One of the most impressive recent computer programs to use a combination of brute-force methods and heuristics to achieve human-level cognitive abilities is IBM’s Watson, a 2,880-processor, 80-teraflop computing behemoth with 15 terabytes of RAM that won a “Jeopardy!” challenge in 2011 against two of the best “Jeopardy!” players in history.³ Now imagine that Watson, having beaten the best humans, began to play against programs like itself but that were more computationally efficient than it was. Watson currently has the ability to learn from its mistakes, and, presumably, future algorithms would further improve its search efficiency. Consequently, there is no reason to believe that better and better brute-force computation would not evolve until it had become, like the brute-force computation that underlies our brains, multilayered, hierarchically organized, contextualized, and highly efficient. That is, the brute-force computation of the future will bear as much resemblance to the brute-force algorithms of today as the computers of today resemble the computers of 1950.

What of the Turing Test in all of this? I am convinced no machine will pass a Turing Test, at least not in the foreseeable future, for the overriding reason I outlined earlier: There will remain recondite reaches of human cognition and physiognomy that will be able to serve as the basis for questions used to trip up any machine. So, set the Turing Test aside. I would be perfectly happy if a machine said to me, “Look, I’m a computer, so don’t ask me any questions that require me to have a body to answer, no stuff about what it feels like to fall off a bicycle or have pins and needles in my foot. This fooling you to think I’m a human is passé. I’m not trying to fool you. I’m a computer, ok? On the oth-

er hand, I'd be happy to discuss, say, romantic poetry, the implications of China's one-child policy, or the current financial crisis, and so on. I can't pass a Turing Test, but so what? I can still be a very interesting conversationalist. I don't need to have actually experienced the pain of hitting my hand with a hammer to talk about it. Just like priests don't need to be married to counsel about-to-be-married young people about married life. They know about the trials and tribulations of married life secondhand by having talked about it all their lives to people who are married. So, while their model of marriage might not be as perfect as the model married people have, it is a good enough approximation to provide real insight about marriage. Think of me in those terms."

Computers of the future, even if they never pass a Turing Test, will potentially be able to see patterns and relationships between patterns that we, with all our experience in the world, might simply have missed. The phenomenal computing capacity of computers, along with ever-better data capture, storage, retrieval, and processing algorithms, has given rise to computer programs that play chess, backgammon, Go, and many other highly "cognitive" games as well, or better, than most humans. They compose music and recognize speech, faces, music, smells, and emotions. Not as well as the best humans—not yet—but this is only the beginning. And just as the early failures of AI contributed to our deeper understanding of the true complexity of human cognition, these programs force us to rethink our anthropocentric ideas on the uniqueness of our cognitive skills. But this rethinking should not be a cause for a concern. That a mass of 100 billion slow and imprecise neurons could organize themselves over the course of many millions of years in such a way as to produce human cognition is an amazing outcome of evolution. However, there is no reason to believe this is the only way to achieve cognition. Understanding human cognition and achieving artificial cognition are two separate endeavors, and, even if each can inform the other, they should not be confused.

The goal of building a machine able to pass a Turing Test will long remain elusive and probably never be achieved. But many other great challenges lie ahead, greater even than flawlessly imitating human cognitive behavior down to the last typing mistake. The degree to which progress can be made in AI will be in direct relation to the degree to which the problems to be solved can be represented cleanly and unambiguously. One of the next great challenges of AI will be the development of computer programs designed to discover and prove elegant new mathematical theorems worthy of publication in mathematics journals, not because they were done by a computer but because the mathematics itself will be worthy of publication. Other challenges will be development of programs that make use of the oceans of data now available to find new relationships between diseases and human behavior or the environment. Yet others will be programs that can look at two different pictures and find their analogous elements.

Conclusion

It is time for the Turing Test to take a bow and leave the stage. The way forward in AI does not lie in an attempt to flawlessly simulate human cognition but in trying to design computers capable of developing their own abilities to understand the world and in interacting with these machines in a meaningful manner. Researchers should be clearer about the distinction between using computers to understand human cognition and using them to achieve artificial cognition, meaning we need to revise our long-held notions of "understanding." Understanding is not something only humans are capable of and, as computers get better at representing and contextualizing patterns, making links to other patterns and analyzing these relationships, we will be forced to concede that they, too, are capable of understanding, even if that understanding is not isomorphic to our own. Few people would argue that interacting with people of other cultures does not enrich our own lives and way of looking at the world. In a similar, if not identical way, in the

not-too-distant future, the same will be true of our interactions with computers.

Acknowledgments

This work was funded in part by a grant from the French National Research Agency (ANR-10-065-GETPI-MA). Thanks to Dan Dennett and, especially, to Melanie Mitchell for their insightful comments on an earlier draft of this article. 

References

1. Bell, G. and Gemmell, J. *Total Recall: How the E-Memory Revolution Will Change Everything*. Dutton, New York, 2009.
2. Dreyfus, H.L. *What Computers Can't Do*. Harper & Row, New York, 1979.
3. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J., Nyberg, E., Prager, J., Schlaefter, N., and Welty, C. Building Watson: An overview of the DeepQA project. *AI Magazine* 31, 3 (2010), 59–79.
4. French, R.M. Dusting off the Turing Test. *Science* 336, 6078 (Apr. 13, 2012), 164–165.
5. French, R.M. Subcognition and the limits of the Turing Test. *Mind* 99, 393 (1990), 53–65.
6. French, R.M. Subcognitive probing: Hard questions for the Turing Test. In *Proceedings of the 10th Annual Cognitive Science Society Conference* (Montréal, July). LEA, Hillsdale, NJ, 1988, 361–367.
7. French, R.M. The Turing Test: The first 50 years. *Trends in Cognitive Sciences* 4, 3 (Mar. 2000), 115–121.
8. Haugeland, J. *Artificial Intelligence, The Very Idea*. MIT Press, Cambridge, MA, 1985.
9. Hayes, P. and Ford, K. Turing Test considered harmful. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (Montréal). Morgan Kaufman Publishers, San Francisco, 1995, 972–977.
10. Hofstadter, D.R. *Metamagical Themas*. Basic Books, New York, 1985, 631–665.
11. Holland, J.H. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975.
12. Kolata, G. How can computers get common sense? *Science* 217, 4566 (Sept. 1982), 1237–1238.
13. Minsky, M. *Computation: Finite and Infinite Machines*. Prentice-Hall, Englewood Cliffs, NJ, 1967.
14. Mitchell, M. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
15. Roy, D. Human Speechome Project, 2011; <http://www.media.mit.edu/press/speechome>
16. Roy, D., Patel, R., DeCamp, P., Kubat, R., Fleischman, M., Roy, B., Mavridis, N., Tellex, S., Salata, A., Guinness, J., Levit, M., and Gorniak, P. The Human Speechome Project. In *Proceedings of the 28th Annual Cognitive Science Society Conference*, R. Sun and N. Miyake, Eds. (Vancouver, B.C., Canada). LEA, Hillsdale, NJ, 2006, 2059–2064.
17. Simon, H. and Newell, A. Heuristic problem solving: The next advance in operations research. *Operations Research* 6, 1 (Jan.–Feb., 1958), 1–10.
18. Talbot, D. A social-media decoder. *Technology Review* (Nov.–Dec. 2011); <http://www.technologyreview.com/computing/38910/>
19. Turing, A.M. Computing machinery and intelligence. *Mind* 59, 236 (1950), 433–460.
20. Whitty, B. The Turing Test: AI's biggest blind alley? In *Machines and Thought: The Legacy of Alan Turing*, P. Millican and A. Clark, Eds. Oxford University Press, Oxford, U.K., 1996, 53–63.

Robert M. French (robert.french@u-bourgogne.fr) is a Research Director of the French National Center for Scientific Research (CNRS) working at the Laboratoire d'Etude de l'Apprentissage et du Développement at the University of Burgundy, Dijon, France.

The challenge of programming molecules to manipulate themselves.

BY DAVID DOTY

Theory of Algorithmic Self-Assembly

SELF-ASSEMBLY IS THE process by which small components automatically assemble themselves into large, complex structures. Examples in nature abound: lipids self-assemble a cell's membrane, and bacteriophage virus proteins self-assemble a capsid that allows the virus to invade other bacteria. Even a phenomenon as simple as crystal formation is a process of self-assembly. How could such a process be described as “algorithmic?” The key word in the first sentence is *automatically*. Algorithms automate a series of simple computational tasks. Algorithmic self-assembly systems automate a series of simple growth tasks, in which the object being grown is simultaneously the machine controlling its own growth.

Although large tracts of the theory presented in this article are applicable to non-molecular systems, much of the motivation arises from nanoscale self-assembly (as an *engineering* field, as opposed to the study of natural self-assembly systems). The broad goal of nanoscience is to manipulate molecules with nanoscale precision. Ambitious long-term applications

include microscopic, chemical-detecting robots that move toward and metabolize pollutants, or the integration of human tissue with an implanted medical device.

Why should computer science have anything to do with nanoscience, beyond the obvious role of developing software-modeling tools? Luca Cardelli, in a panel discussion at the 2011 Conference on DNA Computing and Molecular Programming, observed that while the computing revolution was about the *systematic manipulation of information*, nanoscience is about the *systematic manipulation of matter*. Nanoscience provides a novel justification for studying computation. Many of the traditional forms of manual control are simply not possible at small scales. Automating the growth of molecular structures is not merely faster or more convenient than building such structures by hand. Our hands, and the machines they operate, are simply too large to manipulate individual molecules. We must learn to program molecules to manipulate themselves.

DNA is the molecule of choice in many labs, not for its biological properties but for its information-bearing properties. It is easy to synthesize, and its physical properties are well understood. DNA origami⁴⁵ is currently the most successful laboratory technique for self-assembling DNA. A long scaffold DNA strand is folded into a shape by mixing it with hundreds of shorter staple strands, each of which binds to

» key insights

- The idea of “molecules that can perform computation” is transforming the way we engineer self-assembling molecular systems.
- A small number of simple types of molecules can grow into large, sophisticated nanoscale structures automatically.
- Understanding the fundamental abilities and limitations of these systems is crucial for guiding experimental work. Many known theoretical results have drawn on the theorems, techniques, and paradigms of computer science.



one region of the scaffold with its left half and another region of the scaffold with its right half, bringing the two regions together. A more experimentally challenging technique, DNA tile assembly, is the physical basis for the theoretical work discussed in this article. A DNA tile is a DNA complex with four short single-stranded “sticky” ends protruding from it (see Figure 2a), which are intended to bind the tile to other tiles having sticky ends that are Watson-Crick complementary. Figure 1 shows images of some patterns that have been experimentally assembled with DNA tiles. Figure 2 shows how a DNA tile is modeled as a square with four sides having “specific glues.” DNA origami in its current incarnation is non-algorithmic self-assembly, whereas DNA tile assembly is potentially algorithmic. We now explain this distinction.

Turing⁵⁴ stated, “The ‘computable’ numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.” *Finite means* is formally equated with *algorithm* (for example, a Turing machine, a λ -expression, a Python program, or an appropriately initialized configuration of Conway’s Game of Life). There are uncountably many real numbers but only a countable number of algorithms, so most real

numbers are not computable. What if we wish to make sense of the question, which *integers* are computable? By Turing’s qualitative definition, they are all computable, but it is reasonable to object that some integers (for example, $n = 10^{10,000}$) are easier to compute than others (for example, $m =$ a random sequence of 10,000 digits), in the sense that a much smaller program suffices to compute n than to compute m . To make formal sense of such intuition, the theory of Kolmogorov complexity³³ gives a rigorous *quantitative* measure of the “algorithmic complexity” of an integer (or any other “finite object” such as a finite string or a graph): the length in bits of the shortest algorithm that prints the integer and halts.

Both of these definitions adopt the view that an object is “algorithmically constructed” when there is an algorithm constructing the object that is much smaller than the object itself (infinitely smaller in the former case). From this perspective, *algorithmic self-assembly* describes the self-assembly of a structure whose total number of components (its “size”) is much greater than the number of different *types* of components. Since a component type is reused many times, it has no way of “knowing” where it is going to end up in the structure, so only local infor-

mation is available to guide its attachment. DNA origami therefore does not constitute algorithmic self-assembly, since each staple strand “hardcodes” its position in the final structure. DNA tile assembly, however, has the potential to reuse a small number of tile types to create large structures.

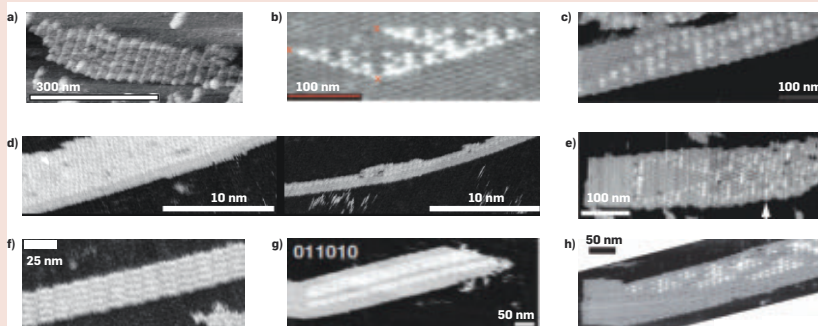
Our combinatorial model of the dynamic behavior of these tiles is called the abstract Tile Assembly Model (aTAM), due to Winfree,⁵⁶ explained briefly in Figure 2. Figure 2c shows seven tile types that fill the entire second quadrant with a painting of the discrete Sierpinski triangle, showing that this pattern is “algorithmically self-assemblable.” The main computation is done by the bottom four rule tile types using *cooperative binding*, which refers to the fact that a tile with only strength-1 glues cannot bind to an assembly unless at least two of them match. (Sometimes this binding strength threshold is called the “temperature” τ , which is usually 2 but not always.) The rule tiles in this example always bind using their south and east glues. Thus the glue labels (bits in this case) can be imagined as inputs to a function computed by the rule tile types (analogous to a transition function in a Turing machine or cellular automaton). The output of the function in this example is the XOR of the bits, which is advertised on the north and west sides.

Parallelism in Molecular Computing: The Bad News

Algorithmic self-assembly is a subfield of molecular computing, highlighted in a news article by Kirk L. Kroeker in the December 2011 issue of *Communications*. The field was initiated in 1994 when Adleman,¹ in a landmark proof-of-concept experiment, designed DNA molecules that interact to solve a 7-vertex case of the Hamiltonian path problem: executing a “DNA algorithm” whose basic operations are well-understood chemical interactions such as hybridization.

Let us state an unequivocal limitation to the power of molecular computing as a model of computation, which applies to algorithmic self-assembly systems as well. Molecular computing is not a magical potion that can be ladled over *NP*-complete problems to transubstantiate them into tractable problems. In its most generic

Figure 1. Experiments with double-crossover tiles.



Atomic force microscopy measures the height of a structure on a surface (mica). Some tile types may have attached hairpins as “labels” (appearing white in images).

- a) Lattice of tiles (single tile type).⁵⁹
- b) Sierpinski triangle in a lattice, no proofreading.⁴⁶
- c) Binary counter in a ribbon, no proofreading.⁶
- d) Standard (left) versus snaked (right) proofreading for suppression of facet errors.¹⁷
- e) Sierpinski triangle in a ribbon, no proofreading.²⁸
- f) Ribbons with zig-zag tiles for suppression of spurious nucleation.⁴⁹
- g) Ribbons nucleating from origami seed, copying a bit string, with proofreading.⁷
- h) Ribbons nucleating from origami seed, executing a binary counter, with partial proofreading.⁷

form, the sorcery of DNA solutions to NP-complete problems proceeds by letting the DNA test all possible solutions in parallel in a single test tube. Assuming a modest solution length of 300 bits, squeezing 2^{300} molecules into a test tube is like installing 2^{300} processors in a parallel computer: there is not enough sand in the world to supply that much silicon.

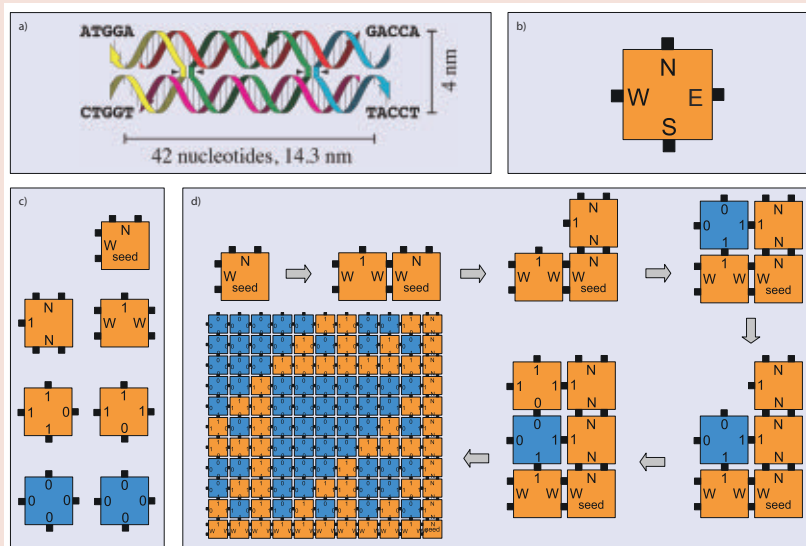
Scientific journals continue to traffic in tales of DNA's exponential search capability. As with Mjölnir, the mountain-crushing hammer of Thor, the time has come to place these tales into the dustbin of mythology, making room for the real—and far more exciting—scientific work that remains to be done in molecular computing. If there is a breakthrough in molecular engineering that will enable the solution of classically intractable problems, it will have to come from quantum mechanics or perhaps from some exotic physical theory yet to be discovered. It will not come about solely by replacing silicon with DNA. But enough pessimism; let's find out what DNA *can* do.

Computational Universality

Winfree⁵⁶ showed that the aTAM is computationally universal, that is, able to simulate any algorithm. This one fact explains the richness of the theory presented in the rest of the article. What exactly does it mean?

The model as stated lacks one feature in common with other computational models: there is no input! One potential way to program a single tile set with different inputs that result in different behaviors is to generalize the idea of a single seed tile type to a larger seed assembly. Within the model, we allow the seed assembly to be any finite stable assembly ("stable" = all cuts of the assembly that separate it into two components must break bonds of total strength at least 2). In practice, seed assemblies are not necessarily made of tiles, but must simply have a perimeter compatible with the tiles, such as a DNA origami shape appropriately augmented with sticky ends on its side (Figure 1g–h). We use the term *tile system* to refer to a finite set of tile types, together with other parameters needed to determine its behavior, such as its seed assembly σ and its temperature τ .

Figure 2. Abstract Tile Assembly Model (aTAM).



- a) Double-crossover tile with four sticky end.
- b) Representation of a tile as a square with sides labeled by string "glues."
- c) Seven tile types. Bond strengths indicated by the number of small black squares on a side: total strength 2 is required to attach a tile to a partially formed assembly of tiles. One tile type is designated as the seed, from which growth is assumed to nucleate.
- d) Growth of the tiles into an assembly with the discrete Sierpinski triangle pattern.

With this convention established, we can now state more formally what it means to claim that the aTAM is computationally universal. For every single-tape Turing machine M , there is a tile set T so that, for every input string x , there is a seed assembly σ of T so that T with seed σ uniquely assembles a space-time configuration transcript of M on input x .^a By "uniquely assembles," we mean that although T with seed σ assembles many different partial assemblies, it has a unique terminal assembly α (terminal = no tile can attach to it). The t^{th} row of α represents the configuration of $M(x)$ at time step t , with σ occupying the first row.

The computational universality of the aTAM implies that arbitrary algorithms may be executed by self-assembling tiles and therefore used to direct the growth of the tiles. However, the aTAM is not just another programming language. The subtle interplay of computation and geometry in self-assembly gives rise to unique characteristics such as the distinction between

computable patterns and self-assembling patterns discussed later.

Modeling Errors

The aTAM is not a realistic model of how DNA tiles actually behave at the molecular level. In particular, it makes two assumptions known not to hold in practice: that tiles never detach from an assembly, and that tiles only attach when their binding strength exceeds the temperature threshold value τ . Winfree⁵⁶ introduced the kinetic Tile Assembly Model (kTAM) as a more realistic model of tile assembly that uses standard laws of chemical kinetics to relax these assumptions. Each tile type is assumed to attach to any binding site on an assembly α , producing a new assembly β , at a rate proportional to its concentration, regardless of its strength of attachment. If we assume that there is only a single seed tile, and that each other tile type is equally concentrated (and much higher count than 1, so that their rate of depletion is essentially 0), then this forward rate is equal for all tile types and approximately constant over time. Call this forward rate r_f . Each tile within β is assumed to detach at a rate proportional

a σ is a $1 \times |x|$ row encoding x in a standard way; we are not cheating by embedding the entire computation $M(x)$ into σ .

to e^{-b} , where $b \in \mathbb{N}$ is the total strength with which the tile is bound. Call this reverse rate $r_{r,b}$. The constant of proportionality depends on factors such as temperature, salinity, and sticky end length. Winfree⁵⁶ showed that by setting these factors and the non-seed tile concentrations such that $r_f = r_{r,2} + \epsilon$ then the probability that the seed grows into a correct assembly (an assembly producible under the aTAM growth rules) approaches 1 as $\epsilon \rightarrow 0$. However, the closer ϵ is to 0, the slower assembly proceeds, with $\epsilon = 0$ representing an unbiased random walk with attachment equally likely as strength 2 detachment.

There is only so much that one can do in a laboratory to precisely control experimental conditions; errors are bound to happen, just as deep-space communication and large-scale data storage inevitably corrupt some of the bits they intend to send or store. What should we do about this? Why not do the same thing that NASA and Google do about it: build in redundancy to help correct the errors! Here, I discuss algorithmic error-correction in the kTAM, showing that under certain conditions, the probability of correct growth can be boosted to enable correct growth under more flexible experimental conditions.

Winfree and Bekbolatov⁵⁸ devised a scheme known as proofreading. It helps correct *growth errors*, shown in Figure 3a, in which an incorrect tile binds with strength 1 where a correct

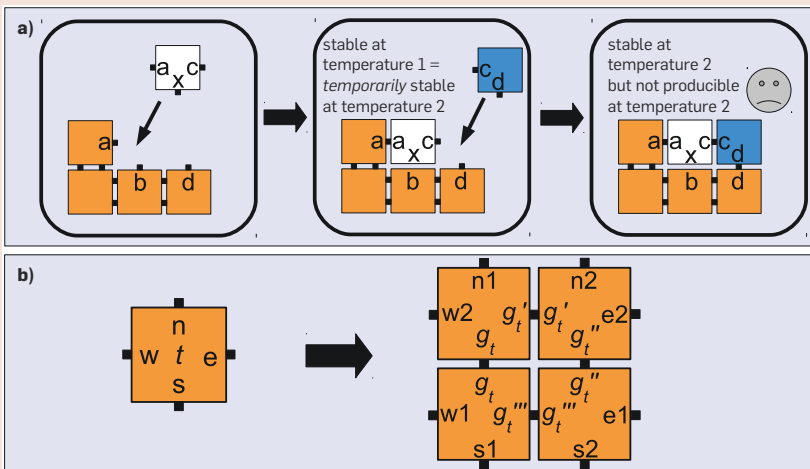
tile could bind with strength 2. A tile bound with only strength 1 may temporarily stick and be secured in place by a subsequent tile attachment. The basic proofreading scheme is shown in Figure 3b. Each tile type t is replaced by a $k \times k$ block of tile types, with glues internal to the block unique to t . This enforces that errors cannot happen in isolation: if there are any errors in the block, then there are $\geq k$ errors. Intuitively, errors happen only slowly (since insufficiently bound tiles fall off more quickly than new tiles attach), so if one error occurs, it is more likely for the erroneous tiles to detach before the block completes than for an additional $k - 1$ errors to occur and secure the block in place. To a very rough approximation, $k \times k$ proofreading changes a base error probability of ϵ to be ϵ^k . This scheme incurs two costs: the number of tile types increases by factor k^2 , and the system suffers a resolution loss since $k \times k$ blocks in the new system represent individual tiles in the original system. Reif, Sahu, and Yin⁴⁴ demonstrated a compact proofreading scheme, in which there is no resolution loss, but in general the tile complexity blowup is exponential. To reduce error probability of ϵ to ϵ^k for tile system T , the new tile system may have up to $|T|^{k^2}$ tile types. Soloveichik and Winfree⁵¹ showed that for all but a restricted class of tile systems, the tile complexity blowup of any compact proofreading scheme is necessarily exponential.

A *facet error* is the attachment of a tile with strength 1 where no tile should go because there is only one adjacent strength 1 glue. Both the proofreading constructions cited^{44,58} correct growth errors but not facet errors. In practice, facet errors tend to dominate the overall behavior. Chen and Goel¹⁴ showed a proofreading scheme, also using a $k \times k$ block replacement as in Winfree-Bekbolatov,⁵⁸ which is resistant to both growth and facet errors, and which achieves error of $O(\epsilon^k)$ on a base error probability of ϵ . In fact, this scheme was experimentally demonstrated¹⁷ to reduce facet errors better than either no error-correction or the Winfree-Bekbolatov proofreading scheme. Chen, Goel, and Luhrs¹⁵ showed that two-dimensional tile systems may be proofread with no resolution loss and only polynomial increase in tile complexity by using the third dimension. It grows a larger structure than the original system, but when projected onto the plane there is no resolution loss. They also introduce combinatorial criteria to help in proving that a proofreading scheme works without needing to carry out the cumbersome Markov process analysis required to analyze kTAM systems.

Doty et al.²⁶ showed a stronger form of error correction using the hierarchical model of self-assembly, in which there is no seed and two assemblies that have assembled independently in parallel are allowed to aggregate together, as opposed to the standard seeded aTAM in which tiles attach one at a time to a growing assembly. In this model, they use the ability of geometry to enforce binding constraints (preventing two assemblies with matching glues from attaching if their shapes are not compatible to allow the glues to touch) to show how to assemble an $n \times n$ square from $O(\log n)$ tile types while guaranteeing the following constraint. Arbitrary strength 1 growth is allowed; however, any assembly that grows sufficiently to become stable at temperature 2 is guaranteed to assemble into the correct final assembly. Thus errors are prevented absolutely, rather than only with high probability.

The errors modeled so far are those of “insufficient attachment:” a tile binds with strength 1 and sticks around long enough to cause prob-

Figure 3. a) Growth error. b) 2×2 Proofreading. Each tile type t is replaced by a $k \times k$ block of tile types, with glues internal to the block unique to t .



lems. We can consider other types of errors. For instance, we consider some tile types to have input and output sides, although this is not enforced from within the model. Consider the following error: after some time, a portion of the assembly is removed to make a hole. This could result in “backward” growth filling in the hole using output glues, possibly resulting in the wrong tiles attaching (this would happen in Figure 2d since XOR is not a 1-1 function). Winfree⁵⁷ introduced a 3×3 block transformation scheme called self-healing that enables the assembly to regrow correctly even when holes are blown out of the assembly, so long as the seed remains present. Soloveichik, Cook, and Winfree⁵⁰ showed how to combine self-healing simultaneously with proof-reading. Chen et al.¹⁶ showed that a more extreme form of self-healing was possible for the particular task of assembling an $n \times n$ square from $O(\log n)$ tile types: the entire square is able to grow from any subassembly of width or height $2 \log n$, without requiring the seed. They also generalize the idea to arbitrary finite shapes (with some resolution loss).

Another type of error is *spurious nucleation*, in which tiles attach to each other without a seed, using strength 1 growth to form a stable assembly from which an incorrect assembly could grow. Schulman and Winfree showed both theoretically⁴⁸ and experimentally⁴⁹ that a certain class of tile systems can be made resistant to nucleation errors, in the sense that with the addition of $O(k)$ extra tile types, in the absence of the seed, the probability is at most $O(2^{-k})$ that sufficiently many tiles aggregate to create a stable assembly from which further growth can occur.

The conclusion to draw from these error correction techniques is that, even though the aTAM does not accurately describe the behavior of DNA tiles, under certain assumptions, it is an implementable “programming language” for tile self-assembly.

Tile Complexity

We have identified the algorithmic aspect of self-assembly with the ability of a small number of tile types to assemble a large structure. This ability will prove crucial to programming large,



Even though the aTAM does not accurately describe the behavior of DNA tiles, under certain assumptions, it is an implementable “programming language” for tile self-assembly.



complex molecular self-assembling systems, since the total number of types of molecular components dominates the cost (in money and time) of implementation.

To quantitatively formalize this idea, Rothemund and Winfree⁴⁷ defined the *tile complexity* of a shape S (a finite, connected subset of $\mathbb{Z}^2$) to be the minimum number of tile types in any tile system—with a single seed tile—that uniquely assembles a single terminal assembly with shape S . The requirement of a size-1 seed avoids cheating by simply letting the seed assembly have the desired shape. They studied the tile complexity of $n \times n$ squares, which has since become a canonical benchmark problem for studying various other aspects of self-assembly, since $n \times n$ squares are in a sense the simplest shape with non-trivial tile complexity. The simplest shape is a single point, whose tile complexity is clearly 1. The next simplest shape might be a $1 \times n$ line. An easy argument shows that its tile complexity is n . Since it is one-dimensional, any stable assembly with that shape must use entirely double-strength glues to hold it together. If fewer than n tile types are used, then one must repeat, but this tile system would then be able to grow an infinite line by repeating the segment between the repetitions. Only in two dimensions is tile complexity nontrivial.

Rothemund and Winfree showed that for most values of n (all algorithmically random n), the tile complexity of an $n \times n$ square is $\Omega(\frac{\log n}{\log \log n})$. Why does this hold? A tile system of size k can be described using $O(k \log k)$ bits ($O(\log k)$ bits per tile type). If the tile system uniquely self-assembles an $n \times n$ square, then that description, together with a constant size aTAM simulator, constitute a program of length $O(k \log k)$ that outputs n . Since for most n , the shortest program outputting n is $\log n$ bits long, we must have $k \log k = \Omega(\log n)$, that is, $k = \Omega(\frac{\log n}{\log \log n})$. They then show that the tile complexity of all $n \times n$ squares is $O(\log n)$, in which $\log n$ tile types each represent a particular bit of n that assemble to form a $1 \times \log n$ row whose north glues represent n in binary. From this assembly, a $O(1)$ tile types attach to assemble an $n \times n$ square, first by growing a counter—essentially a binary-to-unary converter—that counts

from n down to 0, extending the $1 \times \log n$ row into a $n \times \log n$ rectangle, together with $O(1)$ tile types that extend this rectangle into the full square.

The gap between these upper and lower bounds arises from the fact that encoding one bit of n per tile type is wasteful, in an information-theoretic sense. In principle, a set of size k can encode $\log k$ bits per element. Letting $k = \frac{\log n}{\log \log n}$, Adleman et al.² showed how to encode $\approx \log k$ bits of n per tile type, showing the tile complexity of any $n \times n$ square is $O(\frac{\log n}{\log \log n})$. The trick is to encode n in a larger base b . Letting b be the unique power of two satisfying $k \leq b < 2k$, k tile types are required to encode n in base b . An additional $O(k)$ tile types then convert n to base 2, from which the constant set of tile types of Rothemund and Winfree⁴⁷ self-assemble an $n \times n$ square.

Intuitively, most of the complexity of an $n \times n$ square is captured by its width; geometry does not play much of a role. What about shapes with more complicated geometry? One would expect that even if such a shape has a compact algorithmic description, self-assembly that simulates this algorithm may not be possible to execute *within* the shape. Soloveichik and Winfree⁵² showed that in fact, if we ignore scaling factors, then the tile complexity of *every* shape S is closely related to its Kolmogorov complexity $K(S)$, the length in bits of the shortest program outputting the points in the shape. More formally, given a finite shape $S \subset \mathbb{Z}^2$ and positive integer $c \in \mathbb{Z}^+$, let $S^c = \{ (x, y) \mid (\lfloor \frac{x}{c} \rfloor, \lfloor \frac{y}{c} \rfloor) \in S \}$ be S scaled up by factor c . They showed that for every finite shape S , there is a scaling factor c such that the tile complexity of S^c is $\Theta(\frac{K(S)}{\log K(S)})$. This is a tight bound, since no smaller tile system could uniquely self-assemble any scaling of S without contradicting the Kolmogorov complexity of S . The scaling factor c is proportional to the running time of the shortest program for S . A constant set of tile types T produces S^c from a $1 \times \frac{K(S)}{\log K(S)}$ row encoding the program for S . Thus, T is a single tile set that is universally programmable (by seeding with an appropriate program) for building any finite shape S , if scaling factors are ignored.

What is the lesson behind of all these bounds? The ability of tiles to execute algorithms translates directly into



The ability of tiles to execute algorithms translates directly into the ability to assemble large, complex shapes from a very small number of different types of molecular components.



the ability to assemble large, complex shapes from a very small number of different types of molecular components.

Tile complexity has also been studied in many interesting variants of the standard aTAM. Each of these models allows a *single* tile system to be reused for assembling different structures by programming it with different environmental conditions affecting the behavior of the tiles, thus showing that $O(1)$ tile complexity suffices for assembly of complex structures if additional laboratory steps are used.

In the *step assembly* model introduced by Reif,⁴⁵ tile types are added to a test tube in a series of steps, with each step assumed to run to completion before washing away remaining unbound tiles and nonterminal assemblies, before adding new tile types for the next step. Mañuch, Stacho, and Stoll³⁷ showed that for a large class of shapes, including arbitrary shapes scaled by factor 2 or any other shape with a Hamiltonian path, $O(1)$ tile types can assemble the shape in the step assembly model. There is a natural generalization of step assembly known as staged assembly, in which the order of test tube mixing is not monotonic and hierarchical assembly (attachment of two large assemblies to each other) is allowed. A directed graph describes the order of test tube mixing. Under this model, Demaine et al.²⁰ showed that $O(1)$ tile types can be used to assemble any shape, and that certain classes of shapes require only $O(\log n)$ parallel mixing stages (source-to-sink distance in the mixing graph). They also showed a number of trade-offs between this measure (stage complexity) and bin complexity, the total number of test tubes required. In the same model in one dimension, Demaine et al.²¹ showed that for each alphabet Σ , there is a constant set of tile types—each labeled by a symbol from Σ —so that, given any string $x \in \Sigma^*$, the tiles can be mixed to uniquely assemble a linear assembly spelling x . Furthermore, the number of mixing stages required is within a constant of the size of the smallest context-free grammar that produces the singleton language $\{x\}$, if each intermediate stage is required to produce a unique terminal assembly (if not, then there are strings for which more efficient mixings exist).

Another model relaxes the assumption of constant temperature over time: *temperature programming*, in which the temperature is raised and lowered over many stages, with each stage assumed to reach a terminal state before changing the temperature for the next stage. Aggarwal et al.⁵ proved that with one temperature change, a *thin rectangle*—an $n \times k$ rectangle where $k < \frac{\log n}{\log \log n - \log \log \log n}$ —can be assembled from $O(\frac{\log n}{\log \log n})$ tile types, compared to a provable lower bound of $\Omega(\frac{n^k}{k})$ in the standard aTAM. Kao and Schweller³¹ showed that $O(1)$ tile types suffice to assemble any $n \times n$ square using $O(\log n)$ temperature changes. Summers⁵³ showed that $O(1)$ tile types suffices to assemble any shape (scaled up) using temperature programming, using two constructions. In the first, the number of temperature changes is $O(|S|)$, and the scaling factor is $O(1)$. In the second, the number of temperature changes is $O(K(S))$ (the Kolmogorov complexity of S), and the scaling factor is proportional to the running time of the shortest program for S .

Randomized self-assembly uses the inherent nondeterminism in the aTAM in which multiple tile types compete to bind to a single binding site, and tile type concentrations determine the probability of a tile type attaching to a binding site when it competes with others sharing the same input glues. Chandran, Gopalkrishnan, and Reif¹² showed that the tile complexity of a $1 \times n$ line is $\Theta(\log n)$ in the randomized model, that is, $\Theta(\log n)$ tile types are sufficient and necessary to assemble a line of expected length n (compared to n tile types that are provably necessary in the deterministic aTAM). In fact, their tile system is equimolar: all tile types have equal concentrations. Becker, Rapaport, and Rémila⁹ introduced the model of *concentration programming*, in which concentrations of tile types are used to program input to the system. Kao and Schweller³² showed that for each $\delta, \epsilon > 0$, there is a single tile system T so that, for every $n \in \mathbb{Z}^+$, there is a setting of concentrations that cause T to assemble an $n' \times n'$ square, where $(1 - \epsilon)n \leq n' \leq (1 + \epsilon)n$ with probability at least $1 - \delta$. That is, the square is probably approximately assembled. Doty²³ improved this result, showing that for each $\delta > 0$, a single tile system can be used to

assemble an exactly $n \times n$ square for any $n \in \mathbb{Z}^+$ with probability at least $1 - \delta$ using concentration programming.

Demaine et al.²² studied the RNase model (first suggested by Rothmund and Winfree⁴⁷), in which some tile types are assumed to be made of RNA, which is compatible to bind with DNA but has a different chemical structure. An enzyme called ribonuclease—abbreviated RNase—can dissolve RNA but not DNA, so adding RNase after assembly completes allows one to controllably separate the assembly into parts that can now interact with each other. They use this model to improve on the Soloveichik/Winfree⁵² result, showing that for every shape S , the optimal $O(\frac{K(S)}{\log K(S)})$ tile types suffice to assemble a scaled version S^c of S , but with scaling factor $c = O(\log |S|)$, rather than c depending on the running time of S 's shortest program.

In the *flexible glue* model studied by Aggarwal et al.,⁵ unequal glues are allowed to interact with non-zero strength. One could imagine this being implemented, for instance, by multiple sticky ends on a tile's side, in which the strength of interaction depends on the number of sticky ends that match. They showed that with flexible glues, $O(\sqrt{\log n})$ tile types are necessary and sufficient to assemble an $n \times n$ square.

Computational Complexity

Rather than manually analyzing each new shape or class of shapes to determine its tile complexity, it would be beneficial to develop an algorithm that automates the task: given a finite shape S , it outputs the smallest tile system that uniquely assembles S . Unfortunately, an efficient algorithm is unlikely to exist: Adleman et al.³ showed that the problem is *NP*-complete. Here, “uniquely assembles S ” means, assembles one unique terminal assembly, whose shape is S . One could object that a tile system deterministically assembles S if it produces many terminal assemblies that all have the shape S . Under this definition, Bryans et al.¹⁰ showed that the problem, in a sense, is even harder: *NP^{NP}*-complete, that is, *NP*-complete for algorithms that have access to a constant-time subroutine for an *NP*-complete problem such as SAT.

In addition to optimization prob-

lems, there are important verification problems of interest in tile assembly. One easy problem is this: given a tile system T and a (possibly nonterminal) assembly α , can T produce α ? One can simply add tiles to the seed that are consistent with α and have sufficient binding strength until either α is complete, or until no more tiles can be added; the answer is “yes” in the former case and “no” in the latter. A more challenging question: Is α the unique terminal assembly produced by T ? A tile system could produce α through an exponential number of different pathways (orders of tile addition), so it would naïvely seem to require that we check all of them to ensure none of them create a different terminal assembly. However, Adleman et al.³ showed this problem is solvable in quadratic time. On the other hand, in the hierarchical aTAM (in which two large assemblies are allowed to attach), the problem of determining whether α is uniquely assembled is *coNP*-complete.¹¹ This is discouraging, since such decision problems are a formalization of the practically important task of “write a simulator for the hierarchical aTAM.”

Shape building is one goal of self-assembly; another is pattern painting. Briefly, we paint some tile types black (for example, by attaching a streptavidin marker molecule) and say the pattern assembled is the set of positions in the final assembly with a black tile. Such a definition is appropriate for modeling practical goals such as self-assembled circuit layouts, such as those studied by Cook, Winfree, and Rothmund.¹⁹ Although the proof techniques for these verification problems generalize easily to pattern assembly, it remains open to prove or disprove pattern assembly analogues of the optimization results noted here. The optimization problems are uncomputable if the tile system is allowed to grow arbitrarily far outside the pattern, using the same argument that shows Kolmogorov complexity is uncomputable. If the minimum tile system to assemble a given finite pattern could be computed, then this computation could be simulated in the second quadrant by a tile system of size $O(1) + \log n$ searching for the first pattern on the positive x -axis that requires more than n tile

types to assemble, at which point the tile system could grow back to the x -axis assemble that pattern, a contradiction. So to establish (for instance) NP -completeness requires also specifying a bounding box as part of the input and requiring that the assembled shape is a subset of the bounding box. Göös and Orponen³⁰ and Lempinen, Czeizler, and Orponen³⁵ developed branch-and-bound algorithms for a variant of this problem, but these algorithms require exponential time.

One can also consider the assembly of infinite patterns (as in Figure 2d) as the “computability-theoretic” version of self-assembly. Lathrop et al.³⁴ showed that not every computable pattern (subset of $\mathbb{Z}^2$) can be self-assembled, since each time step of computation with tiles irreversibly occupies a point in space, and for some patterns, it requires strictly more time to compute whether that point is part of the pattern. However, with regard to one-dimensional patterns, Patitz and Summers⁴² showed that every computable subset of $\mathbb{N}$ can be self-assembled on the x -axis (crucially using the fact that the assembly occupies a large area outside the x -axis where the actual computation happens). Lathrop et al.³⁴ showed a weaker result for computably enumerable sets: there is a simple quadratically bounded function $f: \mathbb{N} \rightarrow \mathbb{N}$ such that, for every computably enumerable $A \subseteq \mathbb{N}$, $f(A)$ can be self-assembled on the x -axis.

Assembly Time Complexity

Another resource bound to consider is the time required to build a shape (as determined by stochastic chemical kinetics). There is a linear amount of parallelism available for assembling a shape of size N in fewer than N steps. This is because the frontier (set of potential binding sites) of an assembly of size N is potentially as large as $O(N)$, and in a given unit of time, a constant fraction of the frontier will have tiles attach if tile types are sufficiently concentrated. Adleman et al.² showed that any shape of diameter D requires time $\Omega(D)$ to assemble from a deterministic tile system in the aTAM, later shown by Chen and Doty¹³ to hold for nondeterministic systems. Adleman et al.² showed that this lower bound is tight for the assembly of $n \times n$ squares, that is, they can be assem-

bled in time $O(n)$, achievable using the information-theoretically optimal $O(\log n / \log \log n)$ tile types. Reif⁴³ also studied local parallelism within the seeded aTAM, for the purpose of computation rather than shape-building. Reif showed that for many problems such as prefix sum and bitonic merge, it is possible to design a tile system that solves the problem using the optimal parallel speedup. Reif used a generalization of the model known as step assembly, in which tile types are added to a test tube in a series of steps, with each step assumed to run to completion before washing away remaining unbound tiles, before adding new tile types.

In a variant of the model known as the hierarchical aTAM, two assemblies that have assembled independently in parallel are allowed to aggregate together, as opposed to the standard seeded aTAM in which tiles attach one at a time to a growing assembly. The distributed parallelism of the hierarchical aTAM dwarfs the linear local parallelism of the seeded aTAM, so a natural conjecture is that hierarchical assembly could be *much* faster than seeded assembly. Chen and Doty¹³ showed there is a hierarchical tile system with the optimal $O(\frac{\log n}{\log \log n})$ tile types that assembles an $n \times n$ square using $O(\log^2 n)$ parallel stages, which is close to the optimal $\log n$ stages, forming the final $n \times n$ square from four $n/2 \times n/2$ squares, which are themselves recursively formed from $n/4 \times n/4$ squares, and so on. However, despite this nearly maximal parallelism, the system actually requires superlinear time to assemble the square. They extended the definition of *partial order tile systems* studied by Adleman et al. in a natural way to hierarchical assembly and show that no hierarchical partial order tile system can build any shape with diameter N in less than time $\Omega(N)$. Intuitively, although attaching an assembly β of size k to another assembly α increases the size of α by k tiles in one step, a conservation of mass argument shows that the concentration of β is at most $1/k$, so we expect to wait at least k time steps for this attachment to occur. However, Chen and Doty also showed the partial order hypothesis to be necessary, since for infinitely many n , a tile

system can assemble a diameter- $\Theta(n)$ rectangle in time $O(n^{4/5} \log n)$, breaking the linear-time lower bound that applies to all seeded systems and partial order hierarchical systems. It is an open question to determine precisely the extent to which hierarchical assembly imparts a speedup over single-tile assembly.

Intrinsic Universality

Doty et al.²⁴ showed a different notion of universality, with respect to *growth* rather than computation, constructing a single tile set U that simulates any tile system T , by choosing an appropriate seed assembly to encode T using tiles from U . The simulation is intrinsic in the sense that the self-assembly process carried out by U is exactly that carried out by T , with each tile of T represented by an $m \times m$ block of U tiles. Therefore there is a single tile set that can (modulo rescaling) carry out any self-assembly process that can be achieved by any tile system. The challenge is to program all this activity without ever blocking a path that may later be needed for intra-block communication, since no block representing a tile is necessarily aware of the identities of its neighboring blocks, whether they are even present, and if not, whether they will ever arrive.

Power of Cooperative Binding

Cooperative binding is used crucially in all non-trivial constructions in the aTAM. It is also the most challenging aspect of the model to enforce experimentally, so it is reasonable to question its necessity in algorithmic self-assembly. Doty, Patitz, and Summers²⁷ investigated the power of temperature 1 self-assembly in the aTAM, in which all positive glue strengths have sufficient energy to bind a tile, which effectively inhibits the use of cooperative binding. They show that a wide class of deterministic tile systems, satisfying a condition known as *pumpability*, are incapable of universal computation in the sense that they self-assemble only “periodic” (semi-linear) sets. They conjecture that all deterministic, temperature 1 tile systems are pumpable. Mañuch, Stacho, and Stoll³⁸ obtained similar results studying temperature 1 tile systems that assemble finite assemblies and

guarantee no glue mismatches, showing that $\Omega(n)$ tile types are required to assemble shapes of diameter n .

However, with slight changes to the model, universal computation becomes possible. Adleman et al.⁴ devised a nondeterministic temperature 1 tile system that simulates a Turing machine in the following sense: the system has many terminal assemblies, but the unique largest terminal assembly represents the computation. Cook, Fu, and Schweller¹⁸ showed that deterministic tile systems can achieve universal computation if allowed to grow in three dimensions. Patitz, Schweller, and Summers⁴¹ showed that if negative glue strengths are allowed then universal computation is possible, essentially using negative-strength glues to enforce cooperative binding.

But the original question stands: what is the computational power of deterministic, planar, positive-strength, temperature 1 self-assembly? Is cooperative binding truly necessary to compute by planar self-assembly?

The Future

In the early history of computability theory, the optimistic term “elementary” was applied to problems computable in time bounded by a tower of exponentials such as 2^{2^n} . It was not until the 1960s, once faster and faster computers had been built to execute actual algorithms, that polynomial time came to be seen as a more realistic notion of feasibly implementable. Today we confront a similar situation in algorithmic self-assembly, where experimental work has not yet caught up to theory. When it does, the theory will doubtlessly take unanticipated turns. One concrete way that theory can help experiment today is to further develop error correction, such as reducing tile complexity of error-correction constructions or handling broader classes of tile systems than previous constructions.

The complexity of today’s operating systems and other software is made possible only by the development of software engineering techniques to help manage the complexity. Eventually molecular self-assembly systems will scale to the point that no one person could manually keep track of all the types of molecular compo-



Eventually molecular self-assembly systems will scale to the point that no one person could manually keep track of all the types of molecular components and how they are all supposed to fit together.



nents and how they are all supposed to fit together. Software simulators for the aTAM exist^{40,55} but require the user to think at the level of individual tile types, which quickly begins to feel like assembly code programming with a nightmarish instruction set. There is some preliminary work^{8,25} developing higher-level languages for describing aTAM tile systems, but we are far from having a “Python of self-assembly.”

DNA tile assembly is an example of passive self-assembly, in which the tiles are mobile but otherwise stateless. Active self-assembly, using components with a changeable state, may enable sophisticated structures to be built at a lower cost in terms of time, tile complexity, or error-resistance. Majumder, LaBean, and Reif,³⁶ as well as Fujibayashi et al.,²⁹ and Padilla, Liu, and Seeman,³⁹ have proposed theoretical suggestions to augment tiles with a physical mechanism known as strand displacement to enable active signaling across tiles. It remains to see what will actually work in the laboratory, but whatever works will provide a rich theory to explore. As in any nascent field, the most exciting aspect is not knowing what the future holds.

Acknowledgments

I thank Damien Woods, Matt Patitz, and Scott Summers for helpful suggestions.

The author was supported by a Computing Innovation Fellowship under NSF grant 1019343 and NSF grants CCF-1219274 and CCF-1162589, and by the Molecular Programming Project under NSF grant 0832824. 

References

1. Adleman, L.M. Molecular computation of solutions to combinatorial problems. *Science* 266, 5187 (1994), 1021.
2. Adleman, L.M., Cheng, Q., Goel, A. and Huang, M.-D. Running time and program size for self-assembled squares. In *Proceedings of STOC* (2001), 40–748.
3. Adleman, L.M., Cheng, Q., Goel, A. and Huang, M.-D., Kempe, D., de Espanés, P.M. and Rothmund, P.W.K. Combinatorial optimization problems in self-assembly. In *Proceedings of STOC* (2002), 23–32.
4. Adleman, L.M., Kari, J., Kari, L., Reishus, D. and Sosik, P. The undecidability of the infinite ribbon problem: Implications for computing by self-assembly. *SIAM Journal on Computing* 38, 6 (2009), 2356–2381. Preliminary version appeared in FOCS 2002.
5. Aggarwal, G., Cheng, Q., Goldwasser, M.H., Kao, M.-Y., de Espanés, P.M. and Schweller, R.T. Complexities for generalized models of self-assembly. *SIAM Journal on Computing* 34 (2005), 1493–1515. Preliminary version appeared in *Proceedings of SODA* 2004.
6. Barish, R.D., Rothmund, P.W.K. and Winfree, E. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Letters* 5 (2005), 2586–2592.
7. Barish, R.D., Schulman, R., Rothmund, P.W.K. and Winfree, E. An information-bearing seed for nucleating

ACM Transactions on Accessible Computing



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

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



Association for
Computing Machinery

- algorithmic self-assembly. In *Proceedings of the National Academy of Sciences* 106, 15 (Mar. 2009), 6054–6059.
8. Becker, F. Pictures worth a thousand tiles, a geometrical programming language for self-assembly. *Theoretical Computer Science* 410, 16 (2009), 1495–1515.
 9. Becker, F., Rapaport, I. and Rémila, E. Self-assembling classes of shapes with a minimum number of tiles, and in optimal time. In *Proceedings of FSTTCS* (2006), 45–56.
 10. Bryans, N., Chiniforooshan, E., Doty, D., Kari, L. and Seki, S. The power of nondeterminism in self-assembly. *Theory of Computing*. To appear. Preliminary version appeared in *Proceedings of SODA* (2011), 590–602.
 11. Cannon, S., Demaine, E.D., Demaine, M.L., Eisenstat, S., Patitz, M.J., Schweller, R.T., Summers, S.M. and Winslow, A. Two hands are better than one (up to constant factors). Technical Report 1201.1650, Computing Research Repository, 2012.
 12. Chandran, H., Gopalakrishnan, N. and Reif, J.H. The tile complexity of linear assemblies. In *SIAM Journal on Computing* 41, 4 (2012) 1051–1073. Preliminary version appeared in *ICALP* (2009).
 13. Chen, H.-L. and Doty, D. Parallelism and time in hierarchical self-assembly. In *Proceedings of SODA* (2012), 1163–1182.
 14. Chen, H.-L. and Goel, A. Error free self-assembly with error prone tiles. *DNA* 2004.
 15. Chen, H.-L., Goel, A. and Luhrs, C. Dimension augmentation and combinatorial criteria for efficient error-resistant DNA self-assembly. In *Proceedings of SODA* (2008), 409–418.
 16. Chen, H.-L., Goel, A. and Luhrs, C. and Winfree, E. Self-assembling tile systems that heal from small fragments. *DNA* (2007), 30–46.
 17. Chen, H.-L., Schulman, R., Goel, A. and Winfree, E. Reducing facet nucleation during algorithmic self-assembly. *Nano Letters* 7, 9 (Sept. 2007), 2913–2919.
 18. Cook, M., Fu, Y. and Schweller, R.T. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In *Proceedings of SODA* (2011), 570–589.
 19. Cook, M., Rothmund, P. and Winfree, E. Self-assembled circuit patterns. *DNA* (2004), 1979–1979.
 20. Demaine, E.D., Demaine, M.L., Fekete, S.P., Ishaque, M., Rafalin, E., Schweller, R.T. and Souvaine, D.L. Staged self-assembly: Nanomanufacture of arbitrary shapes with $O(1)$ glues. *Natural Computing* 7, 3 (2008), 347–370. Preliminary version appeared in *DNA* (2007).
 21. Demaine, E.D., Eisenstat, S., Ishaque, M. and Winslow, A. One-dimensional staged self-assembly. *DNA* (2011), 100–114.
 22. Demaine, E.D., Patitz, M.J., Schweller, R.T. and Summers, S.M. Self-assembly of arbitrary shapes using RNAse enzymes: Meeting the Kolmogorov bound with small scale factor. In *Proceedings of STACS* (2011).
 23. Doty, D. Randomized self-assembly for exact shapes. *SIAM Journal on Computing*, 39, 8 (2010), 3521–3552. Preliminary version appeared in *Proceedings of FOCS* (2009).
 24. Doty, D., Lutz, J.H., Patitz, M.J., Schweller, R.T., Summers, M. and Woods, D. The tile assembly model is intrinsically universal. In *Proceedings of FOCS* (2012), to appear. IEEE.
 25. Doty, D. and Patitz, M.J. A domain-specific language for programming in the tile assembly model. *Proceedings of DNA* (2009), 25–34.
 26. Doty, D. and Patitz, M.J., Reishus, D., Schweller, R.T. and Summers, S.M. Strong fault-tolerance for self-assembly with fuzzy temperature. In *Proceedings of FOCS* (2010), IEEE, 417–426.
 27. Doty, D. and Patitz, M.J. and Summers, S.M. Limitations of self-assembly at Temperature 1. *Theoretical Computer Science* 412, 1–2 (Jan. 2011), 145–158. Preliminary version appeared in *DNA* (2009).
 28. Fujibayashi, K., Hariadi, R., Park, S.H., Winfree, E. and Murata, S. Toward reliable algorithmic self-assembly of DNA tiles: A fixed-width cellular automaton pattern. *Nano Letters* 8, 7 (2007), 1791–1797.
 29. Fujibayashi, K., Zhang, D., Winfree, E. and Murata, S. Error suppression mechanisms for DNA tile self-assembly and their simulation. *Natural Computing* 8, 3 (2009), 589–612.
 30. Göös, M. and Orponen, P. Synthesizing minimal tile sets for patterned DNA self-assembly. *DNA* (2010), 71–82.
 31. Kao, M.-Y. and Schweller, R.T. Reducing tile complexity for self-assembly through temperature programming. In *Proceedings of SODA* (2006), 571–580.
 32. Kao, M.-Y. and Schweller, R.T. Randomized self-assembly for approximate shapes. In *Proceedings of ICALP* (2008), 370–384.
 33. Kolmogorov, A.N. Three approaches to the quantitative definition of ‘information.’ *Problems of Information Transmission* 1:1 (1965), 7.
 34. Lathrop, J. Lutz, J., Patitz, M. and Summers, S. Computability and complexity in self-assembly. *Theory of Computing Systems* 48 (2011), 617–647. Preliminary version appeared in *CIE* (2008).
 35. Lempiäinen, T., Czeizler, E. and Orponen, P. Synthesizing small and reliable tile sets for patterned DNA self-assembly. *DNA* (2011), 145–159.
 36. Majumder, U., LaBean, T.H. and Reif, J.H. Activatable tiles for compact error-resilient directional assembly. *DNA* (2007), 15–25.
 37. Mañuch, J., Stacho, L. and Stoll, C. Step-assembly with a constant number of tile types. In *Proceedings of ISAAC* (2009), 954–963.
 38. Mañuch, J., Stacho, L. and Stoll, C. Two lower bounds for self-assemblies at Temperature 1. *Journal of Computational Biology* 17, 6 (2010), 841–852.
 39. Padilla, J.E., Liu, W. and Seeman, N.C. Hierarchical self-assembly of patterns from the Robinson tilings: DNA tile design in an enhanced tile assembly model. *Natural Computing* 11, 2 (2012), 323–338.
 40. Patitz, M.J. Simulation of self-assembly in the abstract tile assembly model with ISU TAS. *FNANO* (2009), 209–219.
 41. Patitz, M.J., Schweller, R.T. and Summers, S.M. Exact shapes and Turing universality at Temperature 1 with a single negative glue. *DNA* (2011), 175–189.
 42. Patitz, M.J. and Summers, S.M. Self-assembly of decidable sets. *Natural Computing* 10 (2011), 853–877. Preliminary version appeared in UC 2008.
 43. Reif, J.H. Local parallel biomolecular computation. *DNA* 3 (1999), 217–254.
 44. Reif, J., Sahu, S. and Yin, P. Compact error-resilient computational DNA tiling assemblies. *DNA* 2004.
 45. Rothmund, P.W.K. Folding DNA to create nanoscale shapes and patterns. *Nature* 440, 7082 (2006), 297–302.
 46. Rothmund, P.W.K., Papadakis, N. and Winfree, E. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology* 2, 12 (2004), 2041–2053.
 47. Rothmund, P.W.K. and Winfree, E. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of STOC* (2000), 459–468.
 48. Schulman, R. and Winfree, E. Programmable control of nucleation for algorithmic self-assembly. *SIAM Journal on Computing* 39, 4 (2009), 1581–1616. Preliminary version appeared in *DNA* (2004).
 49. Schulman, R. and Winfree, E. Synthesis of crystals with a programmable kinetic barrier to nucleation. In *Proceedings of the National Academy of Sciences* 104, 39 (2007), 15236–15241.
 50. Soloveichik, D., Cook, M. and Winfree, E. Combining self-healing and proofreading in self-assembly. *Natural Computing* 7, 2 (2008), 203–218.
 51. Soloveichik, D. and Winfree, E. Complexity of compact proofreading for self-assembled patterns. *DNA* (2005).
 52. Soloveichik, D. and Winfree, E. Complexity of self-assembled shapes. *SIAM Journal on Computing* 36, 6 (2007), 1544–1569. Preliminary version appeared in *DNA* (2004).
 53. Summers, S.M. Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. *Algorithmica* 63, 1 (2012) 117–136.
 54. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings of the London Mathematical Society* (1936), 230–265.
 55. Winfree, E. Simulations of computing by self-assembly. Technical Report Caltech CSTR:1998.22. California Institute of Technology, 1998.
 56. Winfree, E. Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology, June 1998.
 57. Winfree, E. Self-healing tile sets. *Nanotechnology: Science and Computing*, Natural Computing Series. J. Chen, N. Jonoska, and G. Rozenberg, eds. Springer, 2006, 55–78.
 58. Winfree, E. and Bekbolatov, R. Proofreading tile sets: Error correction for algorithmic self-assembly. *DNA* (2003), 126–144.
 59. Winfree, E., Liu, F., Wenzler, L.A. and Seeman, N.C. Design and self-assembly of two-dimensional DNA crystals. *Nature* 394, 6693 (1998), 539–544.

David Doty (ddoty@caltech.edu) is a Computing Innovation Fellow and Postdoctoral Scholar in the Department of Computing and Mathematical Sciences at the California Institute of Technology, Pasadena.

© 2012 ACM 0001-0782/12/12

research highlights

P. 90

**Technical
Perspective
High-Level
Data Structures**

By Yannis Smaragdakis

P. 91

An Introduction to Data Representation Synthesis

By Peter Hawkins, Martin Rinard, Alex Aiken, Mooly Sagiv, and Kathleen Fisher

P. 100

**Technical
Perspective
Natural Algorithms
in a Networked
World**

By Ali Jadbabaie

P. 101

Natural Algorithms and Influence Systems

By Bernard Chazelle

Technical Perspective

High-Level Data Structures

By Yannis Smaragdakis

IT IS A defining moment for many a CS student when they start thinking about data structures relationally; when maps, n -tuples, sets, and bags become more natural terms of discourse than lists, vectors, hash tables, and binary trees. I often see a student's eyes light up when realizing that what they need is, say, a mapping from some X s to an unordered set of Y s, with the correctness of the rest of their program unaffected by whether this mapping is implemented by hash tables and lists, trees and vectors, or something else. This difference in thinking is crucial for programming projects with any amount of data structure complexity, from the real world all the way down to university coursework. In a compiler course project, for instance, the difference is striking. Students who manage to grasp the separation of relational and low-level thinking have no trouble adapting to complex requirements. Students who cannot escape their preoccupation with low-level data structures often spend all their time changing the intertwined implementations of a mess of lexically scoped symbol tables, member tables, and global type structures.

This lifting of data structure thinking to the relational level has long inspired computer scientists. Relational databases have given us a vocabulary for data management, but have also hidden the internals of the implementation from the end programmer. General-purpose programming has largely remained low-level. Even when a library offers an abstraction with different implementations—for example, a map that can be implemented via either a hash table or a binary tree—standard relational reasoning is not supported automatically. A map from a pair of X and Y values to Z values is equivalent to a map from X values to maps from Y s to Z s, for instance, but the programmer cannot represent both forms equivalently and has to structure the code for one or the other. As a result, data structure code is almost universally too low-

Hawkins et al. make a contribution to a long line of work in elevating data manipulation.

level to be amenable to compiler analysis, for performance or correctness.

In the following paper, the authors aim at elevating data structure programming to the relational realm. The separation of the relational view from the low-level implementation view of the data structure is performed in a disciplined way, and even automated. Given a relational specification of data to be stored, the system derives and verifies a “decomposition:” a combination of data structures that together implement the relational specification. Insertions, deletions, and lookups on the data structure are performed via actions that remain unchanged for different decompositions. Eventually the system automatically produces code that correctly implements the desired query. Key novelties of the approach are the use of functional dependencies over data and a type system that verifies the adequacy of a decomposition for representing the relations and dependencies.

In a greater context, Hawkins et al. make a contribution to a long line of work in elevating data manipulation. The SETL language⁴ pioneered high-level data structures by offering a programming model based on set theory, together with automatic selection of concrete data structures at compile time. Even closer to the authors' work in terms of programming model, Batory introduced a sequence of specialized languages^{1,2,5} exploring the refine-

ment of a relational specification to concrete, intertwined data structures. The program is written in terms of queries oblivious to the specific data structures used. Once the data structures are selected (manually, in a single configuration statement^{2,5} or automatically via search in the space of data structures¹), a query is classified as a scan, an ordered range lookup, or a direct map lookup, and efficient code is generated for the given structures. Additionally, other interesting angles on the relational view of data structures have been examined, such as the Collection Programming Language (CPL) by Buneman et al.³ CPL offers declarative whole-structure operations—an approach of great power, but probably more difficult to integrate in general-purpose code with imperative updates and a traversal state.

Accordingly, the work toward relational thinking in data structures is not over, nor is it likely to be over soon. The issue of the “right” high-level programming interface is not settled yet. Furthermore, there is always a tension between programmer control and automation. In some cases there may be a benefit to “cracking open the shell” and obtaining better performance via clever low-level data structure manipulations (for example, employing hybrid structures, such as an “augmented” red-black tree). Regardless of where a language or library draws the line, I hope that upon reading this paper you will agree that the effort to conceptually separate relational from low-level data structure design will be key to the future of high-level programming. 

References

1. Batory, D., Chen, G., Robertson, E. and Wang, T. Design wizards and visual programming environments for GenVoca generators. *IEEE Transactions on Software Engineering* 26, 5 (2000), 441–452.
2. Batory, D. and Thomas, J. P2: A lightweight DBMS generator. *Journal of Intelligent Information Systems* 9 (1995), 107–123.
3. Buneman, P., Naqvi, S., Tannen, V. and Wong, L. Principles of programming with complex objects and collection types. *Theoretical Computer Science* 149, 1 (1995), 3–48.
4. Schwartz, J., Dewar, R., Schonberg, E. and Dubinsky, E. *Programming With Sets: An Introduction to SETL*. Springer-Verlag, 1986.
5. Smaragdakis, Y. and Batory, D. DiSTIL: A transformation library for data structures. In *Proceedings of the Conference on Domain-Specific Languages*. USENIX, 1997.

Yannis Smaragdakis (yannis@smaragd.org) is an associate professor in the Department of Informatics at the University of Athens, Greece.

© 2012 ACM 0001-0782/12/12

An Introduction to Data Representation Synthesis

By Peter Hawkins, Martin Rinard, Alex Aiken, Mooly Sagiv, and Kathleen Fisher

Abstract

We consider the problem of specifying combinations of data structures with complex sharing in a manner that is declarative and results in provably correct code. In our approach, abstract data types are specified using relational algebra and functional dependencies. We describe a language of decompositions that permits the user to specify different concrete representations for relations, and show that operations on concrete representations soundly implement their relational specification. We also describe an auto-tuner that automatically identifies the best decomposition for a particular workload. It is easy to incorporate data representations synthesized by our compiler into existing systems, leading to code that is simpler, correct by construction, and comparable in performance to the code it replaces.

1. INTRODUCTION

One of the first things a programmer must do when implementing a system is to commit to particular data structure choices. For example, consider a simple operating system process scheduler. Each process has an ID *pid*, a *state* (running or sleeping), and a variety of statistics such as the *cpu* time consumed. Since we need to find and update processes by ID, we store processes in a hash table indexed by *pid*; as we also need to enumerate processes in each state, we simultaneously maintain a linked list of running processes and a separate list of sleeping processes.

Whatever our choice of data structures, it has a pervasive influence on the subsequent code. Moreover, as requirements evolve it is difficult and tedious to change the data structures to match. For example, suppose we add virtualization support by allowing processes with the same *pid* number to exist in different namespaces *ns*, together with the ability to enumerate processes in a namespace. Extending the existing data structures to support the new requirement may require many changes distributed throughout the code.

Furthermore, invariants on multiple, overlapping data structures that represent different views of the same data are hard to state, difficult to enforce, and easy to get wrong. For the scheduler, we require that each process appears in both the hash table indexed by process ID and exactly one of the running or sleeping lists. Such invariants must be enforced by every piece of code that manipulates the scheduler's data structures. It is easy to forget a case, say by failing to add a process to the appropriate list when it changes state or by failing to delete a hash table entry when

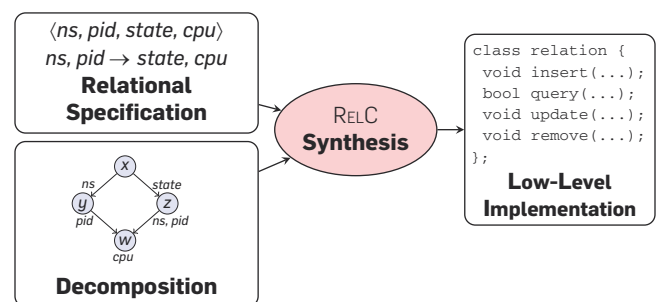
a process terminates. Invariants of this nature require deep knowledge about the heap's structure and are difficult to enforce through existing static analysis or verification techniques.

We propose to use *data representation synthesis*, depicted in Figure 1. In our approach, a data structure client writes code that describes and manipulates data at a high level as *relations*. A data structure designer then provides *decompositions* that describe how those relations should be represented in memory as a combination of primitive data structures. Our compiler RELC takes a relation and its decomposition and synthesizes efficient and correct low-level code that implements the relational interface.

Synthesis allows programmers to describe and manipulate data at a high level as relations, while enabling control of how relations are represented physically in memory. By abstracting data from its representation, programmers no longer prematurely commit to a particular data representation. If programmers want to change or extend their choice of data structures, they need only change the decomposition; the code that uses the relation need not change at all. Synthesized representations are correct by construction; so long as the programmer obeys the constraints listed in the relational specification, invariants on the synthesized data structures are automatically maintained.

Due to space constraints we omit many details from this article. See the full paper¹¹ for a more comprehensive treatment.

Figure 1. Data representation synthesis.



The original version of this paper was published in the *Proceedings of Programming Language Design and Implementation (PLDI)*, 2011, ACM.

2. RELATIONAL ABSTRACTION

We first introduce the relation abstraction via which data structure clients manipulate synthesized data representations. Representing and manipulating data as relations is familiar from databases, and our interface is largely standard. We use relations to abstract a program’s data from its representation. Describing particular representations is the task of the decomposition language of Section 3.

A tuple $t = \langle c_1.v_1, c_2.v_2, \dots \rangle$ maps a set of columns $\{c_1, c_2, \dots\}$ to values v . We write $t(c)$ for the value of column c in tuple t . A relation r is a set of tuples $\{t_1, t_2, \dots\}$ over identical columns C . A relation r has a *functional dependency* $C_1 \rightarrow C_2$ if any pair of tuples in r that is equal on columns C_1 is also equal on columns C_2 .

A *relational specification* is a set of column names C and functional dependencies Δ . For the scheduler example from Section 1, processes may be modeled as a relation with columns $\{ns, pid, state, cpu\}$, where the values of $state$ are drawn from the set $\{S, R\}$, representing sleeping and running processes, respectively. The other columns have integer values. For example, the scheduler might represent three processes as the relation:

$$r_s = \left\{ \langle ns: 1, pid: 1, state: S, cpu: 7 \rangle, \right. \\ \left. \langle ns: 1, pid: 2, state: R, cpu: 4 \rangle, \right. \\ \left. \langle ns: 2, pid: 1, state: S, cpu: 5 \rangle \right\} \quad (1)$$

Not every relation represents a valid set of processes; all meaningful sets of processes satisfy a functional dependency $ns, pid \rightarrow state, cpu$, which allows at most one *state* or *cpu* value for any given process.

2.1. Relational operations

We provide five operations for creating and manipulating relations. Operation `empty ()` creates a new empty relation. The operation `insert  $r$   $t$` inserts tuple t into relation r , `remove  $r$   $s$` removes tuples matching tuple s from relation r , and `update  $r$   $s$   $u$` applies the updates in tuple u to each tuple matching s in relation r . Finally `query  $r$   $s$   $C$` returns the columns C of all tuples in r matching tuple s . The tuples s and u given as arguments to the `remove`, `update`, and `query` operations may be partial tuples, that is, they need not contain every column of relation r . Extending the query operator to handle comparisons other than equality or to support ordering is straightforward; however, for clarity of exposition we restrict ourselves to queries based on equalities.

For the scheduler example, we call `empty()` to obtain an empty relation r . To insert a new running process into r , we invoke:

```
insert r ⟨ns: 7, pid: 42, state: R, cpu: 0⟩
```

The operation

$$\text{query } r \text{ state: } R \{ns, pid\}$$

returns the namespace and ID of each running process in r ,
whereas

```
query r ns: 7, pid: 42 {state, cpu}
```

returns the state and cpu of process 42 in namespace 7. By invoking

update *rs*: 7, *pid*: 42 *state*: S

we can mark process 42 as sleeping, and finally by calling

```
remove rns: 7, pid: 42
```

we can remove the process from the relation.

3. DEFINITION OF DECOMPOSITIONS

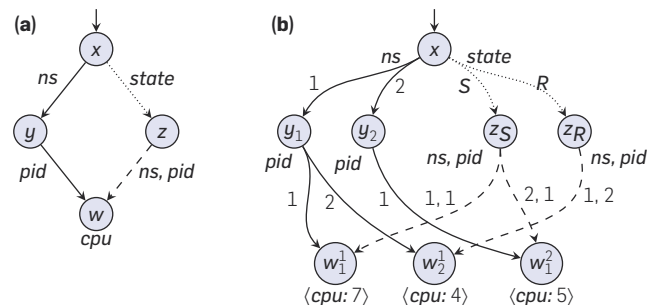
Decompositions describe how to represent relations as a combination of primitive container data structures. Not every decomposition can correctly implement every relation. Our goal in this section is to define *adequacy conditions* which are sufficient conditions for a decomposition to faithfully represent a relation.

3.1. Decompositions

A *decomposition* is a rooted, directed acyclic graph that describes how to represent a relational specification. The subgraph rooted at each node of the decomposition describes how to represent part of the original relation; each edge of the decomposition describes a way of breaking up a relation into a set of smaller relations.

Figure 2(a) shows one possible decomposition for the scheduler relation. Informally, this decomposition reads as follows. From the root (node x), we can follow the left-hand edge, which uses a hash table to map each value n of the ns field to a sub-relation (node y) with the $\{pid, cpu\}$ values for n . From one such sub-relation, the outgoing edge of node y maps a pid (using another hashtable) to a sub-relation consisting of a single tuple with one column, the corresponding cpu time. The $state$ field is not represented on the left-hand path. Alternatively, from the root we can follow the right-hand edge, which maps a process $state$ (running or sleeping) to a sub-relation of the $\{ns, pid, cpu\}$ values of the processes in that state. Each such sub-relation (rooted at node z) maps a

Figure 2. Data representation for a process scheduler: (a) a decomposition, (b) an instance of that decomposition. Solid edges represent hash tables, dotted edges represent vectors, and dashed edges represent doubly linked lists.



$\{ns, pid\}$ pair to the corresponding *cpu* time. While the left path from x to w is implemented using a hash table of hash tables, the right path is a vector with two entries, one pointing to a list of running processes, the other to a list of sleeping processes. Because node w is shared, there is only one physical copy of each *cpu* value, shared by the two access paths.

We now describe the three decomposition primitives.

- A *unit* decomposition, depicted as a node labeled with a set of columns C , represents a single tuple t with columns C . A unit node cannot have any outgoing edges. For example, in Figure 2(a) node w is a unit decomposition containing a single *cpu* value.
- A *map* decomposition, depicted as an edge labeled with a set of columns C , represents a relation as a mapping $\{t \mapsto v_t, \dots\}$ from a set of columns C , called *key columns*, to a set of *residual relations* r_t , one for each valuation t of the key columns. Each residual relation r_t is in turn represented by decomposition v . For example, in Figure 2(a) the edge from y to w labeled *pid* indicates that for each instance of vertex y in a decomposition instance there is a data structure that maps each value of *pid* to a different residual relation, represented using the decomposition rooted at w . Any container that implements a key-value associative map interface can be used to implement the map; the example uses unordered doubly linked lists of key-value pairs, hash tables, and arrays mapping keys to values. The set of containers is extensible; any container implementing a common interface may be used.
- Multiple edges that exit the same node denote a *join*, which represents a relation as the natural join of two sub-relations r_1 and r_2 . Each sub-relation has its own sub-decomposition, allowing a join to represent the same data but with different cost models. In diagrams, join decompositions exist wherever multiple map edges exit the same node. For example, in Figure 2(a) node x has two outgoing edges and hence is the join of two map decompositions.

A *decomposition instance*, or *instance* for short, is a rooted, directed acyclic graph representing a particular relation. Each node of a decomposition corresponds to a set of nodes in an instance of that decomposition. Figure 2(b) shows an instance of the decomposition representing the relation r_s defined in Equation (1). The structure of an instance corresponds to a low-level memory state; nodes are objects in memory and edges are data structures navigating between objects. For example, node $z_{state: S}$ has two outgoing edges, one for each sleeping process; the dashed edge indicates that the collection of sleeping processes is implemented as a doubly linked list.

Each decomposition node has an associated “type,” consisting of a pair of column sets $B \supset C$; every instance of node v in a decomposition instance has a distinct valuation of columns B , and each such instance represents a relation with columns C . In the decomposition of Figure 2(a), the root node x has type $\emptyset \supset \{ns, pid, cpu, state\}$. Since there is only

one valuation for the empty set of columns $\emptyset$, there is exactly one instance of variable x in any instance of the decomposition; further the subgraph of a decomposition instance rooted at x represents a relation with all columns. Similarly, node y has type $\{ns\} \supset \{pid, cpu\}$, implying that there is a distinct instance of node y for each value of the *ns* field in the relation, and that the subgraph rooted at each instance of y represents a relation with columns *pid*, *cpu*. Node z has type $\{state\} \supset \{ns, pid, cpu\}$. Finally node w has type $\{ns, pid, state\} \supset \{cpu\}$.

The structure of a decomposition instance parallels the structure of the decomposition; for each node $v: B \supset C$ in the decomposition the instance contains a corresponding set of node instances $\{v_t, v_t, \dots\}$, each for different valuations of columns B . For example, in Figure 2, decomposition node z has two different instances $z_{state: S}$ and $z_{state: R}$, one for each *state* value in the relation.

3.2. Adequacy of decompositions

We define an *abstraction function* α , defined formally in the full paper,¹¹ which maps decomposition instances to the relation they represent. The abstraction function is defined recursively on the structure of the decomposition instance. Informally, a unit node represents its associated tuple t , a map decomposition represents the union of t joined with $\alpha(v_t)$ for each edge $t \mapsto v_t$ that comprises the map, and a join decomposition represents the natural join of all of its subdecompositions.

Not every relation can be represented by every decomposition. In general a decomposition can only represent relations with specific columns satisfying certain functional dependencies. For example the decomposition $\hat{d}$ in Figure 2(a) cannot represent the relation

$$r' = \left\{ \langle ns: 1, pid: 2, state: S, cpu: 42 \rangle, \right. \\ \left. \langle ns: 1, pid: 2, state: R, cpu: 34 \rangle \right\} \quad (2)$$

since for each pair of *ns* and *pid* values the decomposition $\hat{d}$ can only represent a single value for the *state* and *cpu* fields. However, r does not correspond to a meaningful set of processes—the relational specification in Section 2 requires that all well-formed sets of processes satisfy the functional dependency $ns, pid \rightarrow state, cpu$, which allows at most one *state* or *cpu* value for any given process.

We define *adequacy conditions* that characterize when a decomposition $\hat{d}$ is a suitable representation for a relation with columns C satisfying functional dependencies (FDs). The adequacy conditions are analogous to a type system; if a decomposition $\hat{d}$ is adequate, then it can represent every possible relation with columns C satisfying FDs:

LEMMA 1. *If decomposition $\hat{d}$ is adequate for relations with columns C and FDs, then for each relation r with columns C that satisfies there is some instance d such that $\alpha(d) = r$.*

4. QUERIES AND UPDATES

In Section 3 we introduced decompositions that describe how to represent a relation in memory as a collection of data structures. In this section we show how to compile the relational

operations described in Section 2 into code tailored to a particular decomposition. There are two basic kinds of relational operation, namely queries and mutations. Since we use queries when implementing mutations, we describe queries first.

4.1. Queries and query plans

Recall that the *query* operation retrieves data from a relation; given a relation r , a tuple t , and a set of columns C , a query returns the projection onto columns C of the tuples of r that match tuple t . We implement queries in two stages, similar to a database: *query planning*, which attempts to find the most efficient execution plan q for a query, and *query execution*, which evaluates a particular query plan over a decomposition instance.

In the RELC compiler, query planning is performed at compile time; the compiler generates specialized code to evaluate the chosen plan q with no run-time planning or evaluation overhead. The compiler is free to use any method it likes to choose a query plan, as long as the plan answers the query correctly.

As a motivating example, suppose we want to find the set of *pid* values of processes that match the tuple $ns: 7, state: R$ using the decomposition of Figure 2. That is, we want to find the running processes in namespace 7. One possible strategy would be to look up $state: R$ on the right-hand side, and then to iterate over all ns, pid pairs associated with the state, checking to see whether they are in the correct namespace. Another strategy would be to look up namespace 7 on the left-hand side, and to iterate over the set of *pid* values associated with the namespace. For each *pid*, we then check to see whether the ns and *pid* pair is in the set of processes associated with $state: R$ on the right-hand side. Each strategy has a different computational complexity; the query planner enumerates the alternatives and chooses the “best” strategy.

A *query plan* is a tree of query plan operators. The query plan tree is superimposed on a decomposition and rooted at the decomposition’s root. A query plan prescribes an ordered sequence of nodes and edges of the decomposition instance to visit. There are five query plan operators:

Unit The *qunit* operator returns the unique tuple represented by a unit decomposition instance if that tuple matches t . It returns the empty set otherwise.

Scan The operator *qscan*(q) invokes operator q for each child node v_s where s matches t . Recall a map primitive is a mapping from a set of key columns C to a set of child nodes $\{v_s\}_{s \in C}$. Since operator *qscan* iterates over the contents of a map data structure, it typically takes time linear in the number of entries.

Lookup The *qllookup*(q) operator looks up a particular set of key values in a map decomposition; each of the key columns of the map must be bound in the tuple t given as input to the operator. Query operator q is invoked on the resulting sub-decomposition, if any. The complexity of the *qllookup* depends on the particular choice of data structure. In general, we expect *qllookup* to have better time complexity than *qscan*.

Join The *qjoin*(q_1, q_2, lr) operator performs a join across both sides of a join decomposition. The computational complexity of the join may depend on the order of evaluation. If lr is the value *left*, then first query q_1 is executed on the left side of the join decomposition, and query q_2 is executed on the right side of the join for each tuple returned by tuple q_1 ; the result of the join operator is the natural join of the two subqueries. If lr is the value *right*, the two queries are executed in the opposite order. The *qlr*(q, lr) operator is a special case of the join operator, which performs query q on either the left-hand or right-hand side of a join specified by the argument lr . The other side of the join is ignored.

Recall our motivating example, namely the query

query $r ns: 7, state: R \{pid\}$

that returns the set of running processes in namespace 7. The two plans described above that implement the query are

$$\begin{aligned} q_1 &= \text{qlr}(\text{qllookup}(\text{qscan}(\text{qunit})), \text{right}) \\ q_2 &= \text{qjoin}(\text{qllookup}(\text{qscan}(\text{qunit})), \\ &\quad \text{qllookup}(\text{qllookup}(\text{qunit})), \text{left}). \end{aligned}$$

An important property of the query operators is that they all require only constant space; there is no need to allocate intermediate data structures to execute a query. Constant-space queries can also be a disadvantage; for example, the current restrictions would not allow a “hash-join” strategy for implementing the join operator, nor is it possible to perform duplicate-elimination. It would be straightforward to extend the query language with nonconstant-space operators.

Not every query plan is a correct strategy for evaluating a query. We say that a query plan q is *valid* if q correctly answers queries over a decomposition $\hat{d}$, given a set of input columns A and yielding output columns B . Query validity is defined formally in the full paper.¹¹

To pick good implementations for each query, the compiler uses a query planner that finds the query plan with the lowest cost as measured by a simple heuristic cost estimation function. The query planner enumerates the set of valid query plans for a particular decomposition $\hat{d}$, input columns B , and output columns C , and it returns the plan with the lowest cost. It is straightforward to enumerate query plans, although there may be exponentially many possible plans for a query.

4.2. Mutations

Next we turn our attention to operations *dempty*, which creates an empty instance of a decomposition $\hat{d}$, and *dinsert*, which inserts a tuple t into a decomposition instance d .

To create an empty instance of a decomposition, the *dempty* operation simply creates a single instance of the root node of the decomposition graph; since the relation does not contain any tuples, we do not need to create instances of any map edges. The adequacy conditions for

decompositions ensure that the root node does not contain any unit decompositions, so it is always possible to represent the empty relation.

To insert a tuple t into an instance d of a decomposition $\hat{d}$, for each node $v: B \triangleright C$ in the decomposition we need to find or create a node instance v_s in the decomposition instance, where s is tuple t restricted to columns B . For each edge in the decomposition we also need to find or create an instance of the edge connecting the corresponding pair of node instances. We perform insertion over the nodes of a decomposition in topologically sorted order. For each node v we locate the existing node instance v_s corresponding to tuple t , if any. If no such v_s exists, we create one, inserting v_s into any data structures that link it to its ancestors. For example, inserting the tuple $ns: 2, pid: 1, state: S, cpu: 5$ into the decomposition instance shown in Figure 3(a) yields the state shown in Figure 3(b).

We next consider the dremove and dupdate operations. Operation dremove removes tuples matching tuple s from an instance d of decomposition $\hat{d}$.

To remove tuples matching a tuple t , we compute a cut of the decomposition between those nodes that can only be part of the representation of tuples matching t , and those nodes that may form part of the representation of tuples that do not match t . We then break any edge instances crossing the cut. To find the edge instances to break we can reuse the query planner. Once all such references are removed, the instances of nodes in Y are unreachable from the root of the instance and can be deallocated. We can also clean up any map nodes in X that are now devoid of children.

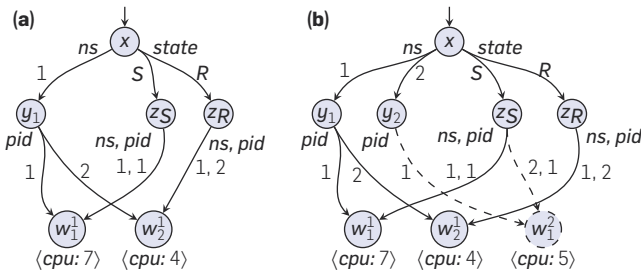
Operation dupdate updates tuples matching s using values from u in an instance d of decomposition $\hat{d}$. Semantically, updates are a removal followed by an insertion. In the implementation we can reuse the nodes and edges discarded in the removal in the subsequent insertion—that is, we can perform the update in place.

4.3. Soundness of relational operations

Finally we show that the operations on decompositions faithfully implement their relational specifications.

THEOREM 2. *Let C be a set of columns, a set of FDs, and $\hat{d}$*

Figure 3. Example of insertion and removal. Inserting the tuple $t = \langle ns: 2, pid: 1, state: S, cpu: 5 \rangle$ into instance (a) produces instance (b); conversely removing tuple t from (b) produces (a). Differences between the instances are shown using dashed lines.



a decomposition adequate for C and $\mathcal{F}$. Suppose a sequence of insert, update and remove operators starting from the empty relation produce a relation r , and each intermediate relation satisfies FDs $\mathcal{F}$. Then the corresponding sequence of dinsert, dupdate, and dremove operators given empty $\hat{d}$ as input produce an instance d such that $\alpha(d) = r$.

5. AUTO-TUNER

Thus far we have concentrated on the problem of compiling relational operations for a particular decomposition of a relation. However, a programmer may not know, or may not want to invest time in finding the best possible decomposition for a relation. We have therefore constructed an *auto-tuner* that, given a program written to the relational interface, attempts to infer the best possible decomposition for that program.

The auto-tuner takes as input a benchmark program that produces as output a cost value (e.g., execution time), together with the name of a relation to optimize. The auto-tuner then exhaustively constructs all decompositions for that relation up to a given bound on the number of edges, recompiles and runs the benchmark program for each decomposition, and returns a list of decompositions sorted by increasing cost. We do not make any assumptions about the cost metric—any value of interest such as execution time or memory consumption may be used.

6. EXPERIMENTS

We have implemented a compiler, named RELC, that takes as input a relational specification and a decomposition, and emits C++ code implementing the relation. We evaluate our compiler using micro-benchmarks and three real-world systems. The micro-benchmarks (Section 6.1) show that different decompositions have dramatically different performance characteristics. Since our compiler generates C++ code, it is easy to incorporate synthesized data representations into existing systems. We apply synthesis to three existing systems (Section 6.2), namely a Web server, a network flow accounting daemon, and a map viewer, and show that synthesis leads to code that is simultaneously simpler, correct by construction, and comparable in performance to the code it replaces.

We chose C++ because it allows low-level control over memory layout, has a powerful template system, and has widely used libraries of data structures from which we can draw. Data structure primitives are implemented as C++ template classes that implement a common associative container API. The set of data structures can easily be extended by writing additional templates and providing the compiler some basic information about the data structure's capabilities. We have implemented a library of data structures that wrap code from the C++ Standard Template Library and the Boost Library, namely doubly linkedlists (`std::list`, `boost::intrusive::list`), binary trees (`std::map`, `boost::intrusive::set`), hash-tables (`boost::unordered_map`), and vectors (`std::vector`). The library includes both nonintrusive containers, in which elements are represented out-of-line as pointers, and intrusive containers, in

which the pointer structure of a container is threaded through the elements of the container with no additional indirection. Since the C++ compiler expands templates, the time and space overheads introduced by the wrappers are minimal.

6.1. Microbenchmarks

We implemented a selection of small benchmarks; here we focus on just one based on directed graphs.

The graph benchmark reads in a directed weighted graph from a text file and measures the times to construct the edge relation, to perform forward and backward depth-first searches over the whole graph, and to remove each edge one-by-one. We represent the edges of a directed graph as a relation `edges` with columns $\{src, dst, weight\}$ and a functional dependency $src, dst \rightarrow weight$. We represent the set of the graph nodes as a relation `nodes` consisting of a single `id` column. The RELC compiler emits a C++ module that implements classes `nodes::relation` and `edges::relation` with methods corresponding to each relational operation. A typical client of the relational interface is the algorithm to perform a depth-first search:

```
edges::relation graph_edges;
nodes::relation visited;
// Code to populate graph_edges elided.

stack<int> stk;
stk.push(v0);
while (!stk.empty()) {
    int v = stk.top();
    stk.pop();
    if (!visited.query(nodes::tuple_id(v))) {
        visited.insert(nodes::tuple_id(v));
        edges::query_iterator_src_dst_weight it;
        graph_edges.query(edges::tuple_src(v), it);
        while (!it.finished()) {
            stk.push(it.output.f_dst());
            it.next();
        }
    }
}
```

The code makes use of the standard STL stack class in addition to an instance of the nodes relation `visited` and an instance of the edges relation `graph_edges`.

To demonstrate the tradeoffs involved in the choice of decomposition, we used the auto-tuner framework to evaluate three variants of the graph benchmark under different decompositions. We used a single input graph representing the road network of the northwestern USA, containing 1,207,945 nodes and 2,840,208 edges. We used three variants of the graph benchmark: a forward depth-first search (DFS); a forward DFS and a backward DFS; and finally a forward DFS, a backward DFS, and deletion of all edges one at a time. We measured the elapsed

time for each benchmark variant for the 84 decompositions that contain at most 4 map edges (as generated by the auto-tuner).

Timing results for decompositions that completed within an 8s time limit are shown in Figure 4. Decompositions that are isomorphic up to the choice of data structures for the map edges are counted as a single decomposition; only the best timing result is shown for each set of isomorphic decompositions. There are 68 decompositions not shown that did not complete any of the benchmarks within the time limit. Since the auto-tuner exhaustively enumerates all possible decompositions, naturally only a few of the resulting decompositions are suitable for the access patterns of this particular benchmark; for example, a decomposition that indexes edges by their weights performs poorly.

Figure 5 shows three representative decompositions from those shown in Figure 4 with different performance characteristics. Decomposition 1 is the most efficient for forward traversal; however, it performs terribly for backward traversal since it takes quadratic time to compute predecessors. Decompositions 5 and 9 are slightly less efficient for forward traversal, but are also efficient for

Figure 4. Elapsed times for directed graph benchmarks for decompositions up to size 4 with identical input. For each decomposition we show the times to traverse the graph forward (F), to traverse both forward and backward (F+B), and to traverse forward, backward, and delete each edge (F+B+D). We elide 68 decompositions that did not finish a benchmark within 8 s.

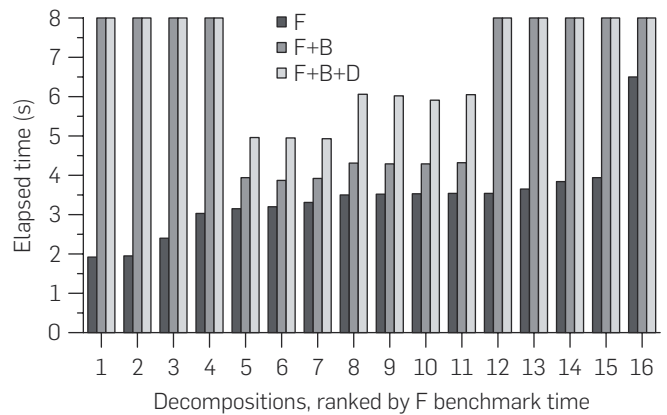
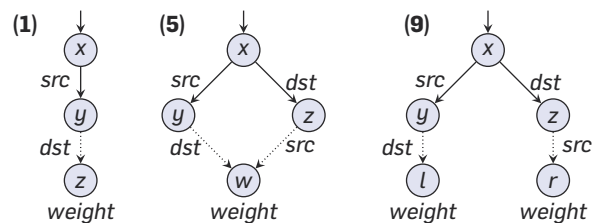


Figure 5. Decompositions 1, 5, and 9 from Figure 4. Solid edges represent instances of `boost::intrusive::map`, dotted edges represent instances of `boost::intrusive::list`.



backward traversal, differing only in the sharing of objects between the two halves of the decomposition. The node sharing in decomposition 5 is advantageous for all benchmarks since it requires fewer memory allocations and allows more efficient implementations of insertion and removal; in particular because the lists are intrusive the compiler can find node w using either path and remove it from both paths without requiring any additional lookups.

6.2. Synthesis in existing systems

To demonstrate the practicality of our approach, we took three existing open-source systems—thttpd, Ipcap, and ZTopo—and replaced core data structures with relations synthesized by RELC.

The thttpd Web server is a small and efficient Web server implemented in C. We reimplemented the module of thttpd that caches the results of the `mmap()` system call. When thttpd receives a request for a file, it checks the cache to see whether the same file has previously been mapped into memory. If a cache entry exists, it reuses the existing mapping; otherwise it creates a new mapping. If the cache is full, then the code traverses through the mappings removing those older than a certain threshold.

The IpCap daemon is a TCP/IP network flow accounting system implemented in C. IpCap runs on a network gateway, and counts the number of bytes incoming and outgoing from hosts on the local network, producing a list of network flows for accounting purposes. For each network packet, the daemon looks up the flow in a table, and either creates a new entry or increments the byte counts for an existing entry. The daemon periodically iterates over the collection of flows and outputs the accumulated flow statistics to a log file; flows that have been written to disk are removed from memory. We replaced the core packet statistics data structures with relations implemented using RELC.

ZTopo is a topographic map viewer implemented in C++. A map consists of millions of small image tiles, retrieved using HTTP over the internet and reassembled into a seamless image. To minimize network traffic, the viewer maintains memory and disk caches of recently viewed map tiles. When retrieving a tile, ZTopo first attempts to locate it in memory, then on disk, and as a last resort over the network. The tile cache was originally implemented as a hash table, together with a series of linked lists of tiles for each state to enable cache eviction. We replaced the tile cache data structure with a relation implemented using RELC.

Table 1 shows non-comment lines of code for each test case. In each case the synthesized code is comparable to or shorter than the original code in size. Both the thttpd and ipcap benchmarks originally used open-coded C data structures, accounting for a large fraction of the decrease in line count. ZTopo originally used C++ STL and Boost data structures, so the synthesized abstraction does not greatly alter the line count. The ZTopo benchmark originally contained a series of fairly subtle dynamic assertions that verified that the two different representations of a tile's state were in agreement; in the synthesized version the compiler

automatically guarantees these invariants, so the assertions were removed.

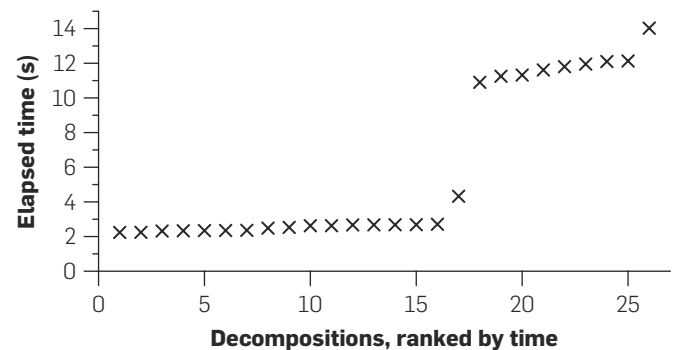
For each system, the relational and non-relational versions had equivalent performance. If the choice of data representation is good enough, data structure manipulations are not the limiting factor for these particular systems. The assumption that the implementations are good enough is important, however; the auto-tuner considered plausible data representations that would have resulted in significant slow-downs, but found alternatives where the data manipulation was no longer the bottleneck. For example we used the auto-tuner on the Ipcap benchmark to generate all decompositions up to size 4; Figure 6 shows the elapsed time for each decomposition on an identical random distribution of input packets. The best decomposition is a binary-tree mapping local hosts to hash-tables of foreign hosts, which performs approximately $5\times$ faster than the decomposition ranked 18th, in which the data structures are identical but local and foreign hosts are transposed. For this input distribution the best decomposition performs identically to the original handwritten code.

Our experiments show that different choices of decomposition lead to significant changes in performance (Section 6.1), that we can tune the representation based on evolving workloads, and that the best performance is comparable to existing handwritten implementations (Section 6.2). The resulting code is concise (Sections 6.1 and 6.2), and the

Table 1. Non-comment lines of code for existing system experiments. For each system, we report the size of entire original system and just the source module we altered, together with the size of the altered source module and the mapping file when using synthesis.

System	Original		Synthesis	
	Total	Module	Decomposition	Module
thttpd	7050	402	42	239
Ipcap	2138	899	55	794
ZTopo	5113	1083	39	1048

Figure 6. Elapsed time for IpCap to log 3×10^5 random packets for 26 decompositions up to size 4 generated by the auto-tuner, ranked by elapsed time. The 58 decompositions not shown did not complete within 30 s.



soundness of the compiler (Theorem 2) guarantees that the resulting data structures are correct by construction.

7. DISCUSSION AND RELATED WORK

7.1. Databases and relational programming

Literature on lightweight databases also advocates a programming model based on relations, which are implemented in the backend using container data structures.^{2,3,7,22} A novel aspect of our approach is that our relations can have specified restrictions (specifically, functional dependencies). These restrictions, together with the fact that in our compilation context the set of possible queries is known in advance, enable a wider range of possible implementations than a traditional database, particularly representations using complex patterns of sharing. Also new in our work is the notion of adequate decompositions and a proof that operations on adequate decompositions are sound with respect to their relational specifications. Our definition of adequacy is informed in particular by the classic Boyce-Codd Normal Form in relational schema design.

Unlike previous lightweight database work, we describe a dynamic auto-tuner that can automatically synthesize the best decomposition for a particular relation, and we present our experience with a full implementation of these techniques in practice. The auto-tuner framework has a similar goal to AutoAdmin.⁶ AutoAdmin takes a set of tables, together with a distribution of input queries, and identifies a set of indices that are predicted to produce the best overall performance under the query optimizer's cost model. The details differ because our decomposition and query languages are unlike those of a conventional database.

Many authors propose adding relations to both general- and special-purpose programming languages.^{5, 17, 19, 23} We focus on the orthogonal problem of specifying and implementing the underlying representations for relational data. Data models such as E/R diagrams and UML also rely heavily on relations. One potential application of our technique is to close the gap between modeling languages and implementations.

7.2. Synthesizing data representations

The problem of automatic data structure selection was explored in SETL²⁰ and has also been pursued for Java collection implementations.²¹ The SETL representation sublanguage⁸ maps abstract SETL set and map objects to implementations, although the details are quite different from our work. Unlike SETL, we handle relations of arbitrary arity, using functional dependencies to enforce complex sharing invariants. In SETL, set representations are dynamically embedded into carrier sets under the control of the runtime system, while by contrast our compiler synthesizes low-level representations for a specific decomposition with no runtime overhead.

Synthesizing specialized data representations has previously been considered in other domains. Ahmed et al.¹ proposed transforming dense matrix computations into implementations tailored to specific sparse representations as a technique for handling the proliferation of complicated sparse representations.

7.3. Synthesis versus verification approaches

A key advantage of data representation synthesis over handwritten implementations is the synthesized operations are correct by construction, subject to the correctness of the compiler. We assume the existence of a library of data structures; the data structures themselves can be proved correct using existing techniques.²⁴ Our system provides a modular way to assemble individually correct data structures into a complete and correct representation of a program's data.

The Hob system uses abstract sets of objects to specify and verify end-to-end properties of systems using multiple data structures that share objects.¹⁴ Monotonic typestates enable aliased objects to monotonically change their typestates in the presence of sharing without violating type safety.¹⁰ Researchers have developed systems that have mechanically verified data structures (e.g., hash tables) that implement binary relational interface.²⁴ The relation implementation presented in this paper is more general (it can implement relations of arbitrary arity) and solves problems orthogonal to those addressed in previous research.

7.4. Specifying and inferring shared representations

The decomposition language provides a “functional” description of the heap that separates the problem of modeling individual data structures from the problem of modeling the heap as a whole. Unlike previous work, decompositions allow us to state and reason about complex sharing invariants that are difficult to state and impossible to verify using previous techniques. Separation logic allows elegant specifications of disjoint data structures,¹⁸ and mechanisms have been added to separation logic to express some types of sharing.^{4,9} Some static analysis algorithms infer some sharing between data structures in low level code^{12, 13, 15, 16}, however, verifying overlapping shared data structures in general remains an open problem for such approaches. The combination of relations and functional dependencies allows us to reason about sharing that is beyond current static analysis techniques.

8. CONCLUSION

We have presented a system for specifying and operating on data at a high level as relations while correctly compiling those relations into a composition of low-level data structures. Most unusual is our ability to express, and prove correct, the use of complex sharing in the low-level representation. We show using three real-world systems that data representation synthesis leads to code that is simpler, correct by construction, and comparable in performance to existing handwritten code.

Acknowledgments

This work was supported by NSF grants CCF-0702681 and CNS-050955. The fourth author thanks Viktor Kuncak, Patrick Lam, Darko Marinov, Alex Sălcianu, and Karen Zee for discussions in 2001–2002 on programming with relations, with the relations implemented by automatically generated linked data structures. The second author thanks

Daniel S. Wilkerson and Simon F. Goldsmith for discussions in 2005 on Wilkerson's language proposal Orth, which includes relational specifications of data structures, the generation of functions for querying and maintaining them, and was further envisioned by Goldsmith to perform automatic data structure selection via profiling. 

References

- Ahmed, N., Mateev, N., Pingali, K., Stodghill, P. A framework for sparse matrix code synthesis from high-level specifications. In *Supercomputing* (Nov. 2000), IEEE Computer Society, 58.
- Batory, D., Chen, G., Robertson, E., Wang, T. Design wizards and visual programming environments for GenVoca generators. *IEEE Trans. Softw. Eng.* 26, 5 (2000), 441–452.
- Batory, D., Thomas, J. P2: A lightweight DBMS generator. *J. Intell. Inform. Syst.* 9 (1997), 107–123.
- Berdine, J., Calgano, C., Cook, B., Distefano, D., O'Hearn, P.W., Wies, T., Yang, H. Shape analysis for composite data structures. In *CAV* (2007), volume 4590 of *LNCS*. Springer, Berlin/Heidelberg, 178–192.
- Bierman, G., Wren, A. First-class relationships in an object-oriented language. In *ECOOP* (2005), volume 3586 of *LNCS*. Springer, Berlin/Heidelberg, 262–286.
- Chaudhuri, S., Narasayya, V.R. An efficient cost-driven index selection tool for Microsoft SQL Server. In *VLDB* (1997), Morgan Kaufmann Publishers, San Francisco, CA, 146–155.
- Cohen, D., Campbell, N. Automating relational operations on data structures. *IEEE Softw.* 10, 3 (May 1993), 53–60.
- Dewar, R.B.K., Grand, A., Liu, S.C., Schwartz, J.T., Schonberg, E. Programming by refinement, as exemplified by the SETL representation sublanguage. *ACM Trans. Program. Lang. Syst.* 1, 1 (Jan. 1979), 27–49.
- Distefano, D., Parkinson, M.J. jStar: towards practical verification for Java. In *OOPSLA* (2008), ACM, New York, NY, 213–226.
- Fähndrich, M., Leino, K.R.M. Heap monotonic typestates. In *International Workshop on Alias Confinement and Ownership* (Jul. 2003).
- Hawkins, P., Aiken, A., Fisher, K., Rinard, M., Sagiv, M. Data representation synthesis. In *PLDI* (2011), 38–49.
- Kreiker, J., Seidl, H., Vojdani, V. Shape analysis of low-level C with overlapping structures. In *VMCAI* (2010), volume 5044 of *LNCS*. Springer, Berlin/Heidelberg, 214–230.
- Kuncak, V., Lam, P., Rinard, M. Role analysis. In *POPL* (2002), ACM, New York, NY, 17–32.
- Lam, P., Kuncak, V., Rinard, M. Generalized typestate checking for data structure consistency. In *VMCAI* (2005), volume 3385 of *LNCS*. Springer, Berlin/Heidelberg, 430–447.
- Lee, O., Yang, H., Petersen, R. Program analysis for overlaid data structures. In *CAV* (2011), volume 6806 of *LNCS*. Springer, Berlin/Heidelberg, 592–608.
- McCloskey, B., Reps, T., Sagiv, M. Statically inferring complex heap, array, and numeric invariants. In *Static Analysis* (2011), volume 6337 of *LNCS*. Springer, Berlin/Heidelberg, 71–99.
- Meijer, E., Beckman, B., Bierman, G. LINQ: Reconciling objects, relations and XML in the .NET framework. In *SIGMOD* (2006), ACM, New York, NY, 706–706.
- Reynolds, J.C. Separation logic: A logic for shared mutable data structures. In *LICS* (2002), 55–74, invited paper.
- Rothamel, T., Liu, Y.A. Efficient implementation of tuple pattern based retrieval. In *PEPM* (2007), ACM, New York, NY, 81–90.
- Schonberg, E., Schwartz, J.T., Sharir, M. Automatic data structure selection in SETL. In *POPL* (1979), ACM, New York, NY, 197–210.
- Shacham, O., Vechev, M., Yahav, E. Chameleon: adaptive selection of collections. In *PLDI* (2009), ACM, New York, NY, 408–418.
- Smaragdakis, Y., Batory, D. DiStiL: a transformation library for data structures. In *Conference on Domain-Specific Languages (DSL '97)* (Oct. 1997), USENIX, 257–271.
- Vaziri, M., Tip, F., Fink, S., Dolby, J. Declarative object identity using relation types. In *ECOOP* (2007), volume 4609 of *LNCS*. Springer, Berlin/Heidelberg, 54–78.
- Zee, K., Kuncak, V., Rinard, M. Full functional verification of linked data structures. In *PLDI* (2008), ACM, New York, NY, 349–361.

The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or U.S. Government.
Contact Author: Peter Hawkins (hawkinsp@cs.stanford.edu)

Peter Hawkins (hawkinsp@cs.stanford.edu), Stanford University.

Mooly Sagiv (msagiv@post.tau.ac.il), Tel-Aviv University.

Martin Rinard (rinard@csail.mit.edu), MIT.

Kathleen Fisher (kfisher@eecs.tufts.edu), Tufts University.

Alex Aiken (aiken@cs.stanford.edu), Stanford University.

© 2012 ACM 0001-0782/12/12

World-Renowned Journals from ACM

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

ACM Transactions on Interactive Intelligent Systems



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

ACM Transactions on Computation Theory



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

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



Association for
Computing Machinery

Advancing Computing as a Science & Profession

www.acm.org/pubs

Technical Perspective

Natural Algorithms in a Networked World

By Ali Jadbabaie

HOW DO BIRDS flock and fish school? How do individuals in a social network reach agreement, even though they are often only influenced by other like-minded individuals? How do fireflies flash synchronously? How can one engineer a swarm of robots to behave like bird flocks?

These are some of the questions that have occupied the minds of many researchers across a diverse set of disciplines over the past two decades, from computer graphics to statistical physics to evolutionary biology, and from control theory to robotics, mathematics, and computer science. Most of the early research, which happened in computer graphics and statistical physics, was on modeling and simulation of collective behavior. Over the past decade the focus has shifted to rigorous theory, leading to what one might call a theory of collective phenomena in multi-agent systems. This theory blends dynamical systems, graph theory, Markov chains, and algorithms.

The collective phenomena are often modeled as many-degrees-of-freedom (discrete-time or continuous time) dynamical systems with an additional twist that the interconnection structure between individual dynamical systems change, since the motion of each node in a flock (or opinion of an individual) is affected primarily by those in the node's *local neighborhood*. The twist here is that the local neighborhood is not fixed: neighbors are defined based on the actual state of the system: For example, in case of opinion dynamics, sociological models indicate that individuals are often influenced by those whose opinions are close to them. In other words, as opinions evolve, neighborhood structures change as the function of the evolving opinion, resulting in a catch 22. Similarly, as birds aggregate their directions with their neighbors, the set of their neighbors change.

When the neighborhood structures

stay intact throughout the evolution of the dynamics, the behavior of the model is easy to analyze. Using Peron-Frobenius theory, one can easily show that connectivity of the network induced by the local neighborhood relations is in fact enough to result in global agreement (in opinions, direction of motion in a flock, or frequency of oscillations in coupled oscillators). One can extend the analysis to the case of time varying or switching networks. If the network change is modeled exogenously, in order to get agreement all that is needed is to make sure the underlying graphs that encode the neighborhood structure stay *connected over time*, that is, over time the union of the edges of the resulting graphs corresponds to a connected graph for all times. The focus on the case where the neighborhood change is exogenous is not without its merits, but this is mostly done for convenience. How can we take the explicit dependence of the graph on the state of the evolving dynamical system into account? This is where the following paper by Bernard Chazelle makes a major contribution.

This paper is a unique exception in the literature, where the endogenous change is explicitly taken into account, leading to the beginnings of a general theory of *influence systems*: where a dynamical system sits on top of nodes of a graph and the edges of the graph change endogenously as the function of the node states. What is the behavior of this (undoubtedly complex) dynamical system? What tools do we need to analyze such systems? These are the questions the paper aims to address. As it turns out, even in the simplest of cases (where the graph is a geometric graph in one dimension and the update is piecewise linear) this is not an easy question to answer.

First, Chazelle introduces us to the notion of the *s-Energy*: A param-

eterized family of Lyapunov functions that represent the evolution of global misalignment between flockmates. Via a tour de force roller coaster ride through dynamical systems, computational geometry, combinatorics, complexity theory, and algorithms, Chazelle creates an “algorithmic calculus” for *diffusive influence systems*: He shows us surprisingly, that the orbit or flow of such systems are attracted to a fixed point in the bidirectional case, and a limit cycle for almost all arbitrarily small random perturbations. But wait, there is more: the convergence time can also be bounded in both cases and the bounds are essentially optimal.

The impressive setup of the diffusive influence system developed by Chazelle creates a near universal setup for analyzing various problems involving collective behavior in networked multi-agent systems, from flocking, opinion dynamics, information aggregation, to synchronization problems. This is as close as one can get to a domain-independent analysis of such networked systems. In short, we need more tools and methods like this to study *natural algorithms* in an increasingly networked world. To make progress on natural algorithms, we might need to rethink the siloed view of education as partitioned into current academic departments. Chazelle's work shows that we can no longer afford to separate algorithms, complexity, combinatorics, and graphs from systems theory and dynamical systems. **C**

Ali Jadbabaie is a professor of Electrical and Systems Engineering at the University of Pennsylvania with secondary appointments in the departments of Computer and Information Science and Operations and Information Management.

Natural Algorithms and Influence Systems

By Bernard Chazelle

Abstract

Algorithms offer a rich, expressive language for modelers of biological and social systems. They lay the grounds for numerical simulations and, crucially, provide a powerful framework for their analysis. The new area of *natural algorithms* may reprise in the life sciences the role differential equations have long played in the physical sciences. For this to happen, however, an “algorithmic calculus” is needed. We discuss what this program entails in the context of *influence systems*, a broad family of multiagent models arising in social dynamics.

1. INTRODUCTION

The gradual elevation of “computational thinking” within the sciences is enough to warm the heart of any computer scientist. Yet the long-awaited dawning of a new age may need to wait a little longer if we cannot move beyond the world of simulation and build a theory of *natural algorithms* with real analytical heft. By “natural algorithms,” I mean the myriad of algorithmic processes evolved by nature over millions of years. Just as differential equations have given us the tools to explain much of the physical world, so will natural algorithms help us model the living world and make sense of it. At least this is the hope and, for now, I believe, one of the most pressing challenges facing computer science.

1.1. Science or engineering?

To draw a fine line between science and engineering is a fool’s errand. Unrepentant promiscuity makes a clean separation neither wise nor easy. Yet a few differences bear mentioning. If science is the study of the nature we have, then engineering is the study of the nature we want: the scientist will ask how the valley was formed; the engineer will ask how to cross it. Science is driven by curiosity and engineering by need: one is the stuff of discovery, the other of invention. The path of science therefore seems more narrow. We want our physical laws to be right and our mousetraps to be useful. But there are more ways to be useful than to be right. Engineering can “negotiate” with nature in ways science cannot. This freedom comes at a price, however. Any mousetrap is at the mercy of a better one. PageRank one day will go; the Second Law of thermodynamics never will. And so algorithms, like mousetraps, are human-designed tools: they are engineering artifacts.

Or are they? Perhaps search engines do not grow on trees, but leaves do, and a sophisticated algorithmic formalism, *L-systems*, is there to tell us how²⁰. It is so spectacularly accurate, in fact, that the untrained eye will struggle to pick out computer-generated trees from the real thing. The algorithmic modeling of bird flocking has been no less successful.

Some will grouch that evolution did not select the human eye for its capacity to spot fake trees and catch avian impostors. Ask a bird to assess your computer-animated flock, they will snicker, and watch it cackle with derision. Perhaps, but the oohs and ahhs from fans of CGI films everywhere suggest that these models are on to something. These are hardly isolated cases. *Natural algorithms* are quickly becoming the language of choice to model biological and social processes. And so algorithms, broadly construed, are *both* science and engineering.

1.2. It is all about language

The triumph of 20th-century physics has been, by and large, the triumph of mathematics. A few equations scattered on a single page of paper explain most of what goes on in the physical world. This miracle speaks of the organizing principles of the universe: symmetry, invariance, and regularity—precisely the stuff on which mathematics feasts. Alas, not all of science is this tidy. Biology = physics + history; but history is the great, unforgiving symmetry breaker. Instead of identical particles subject to the same forces, the life sciences feature autonomous agents, each one with its own idea of what laws to obey. It is a long way, scientifically speaking, from planets orbiting the sun in orderly fashion to unruly slime molds farming bacterial crops. Gone are the symmetry, invariance, and clockwork regularity of astronomy: what we have is, well, sludge. But the sludge follows a logic that has its own language, the language of *natural algorithms*.

The point of departure with classical mathematics is indeed linguistic. While differential equations are the native idiom of electromagnetism, no one believes that cancer has its own “Maxwell’s equations.” Yet it may well have its own natural algorithm. The chain of causal links, some deterministic, others stochastic, cannot be expressed solely in the language of differential equations. It is not just the diversity of factors at play (genetic, infectious, environmental, etc.); nor is it only their heterogeneous modes of interaction. It is also the need for a narrative of collective behavior that can be expressed at different levels of abstraction: first-principles; phenomenological; systems-level; and so on. The issue is not size alone: the 3-body problem may be intractable but, through the magic of *universality*, intricate phase transitions among 10^{30} particles can be predicted with high accuracy. The promise of agent-based natural algorithms is to deliver tractable abstractions for descriptively complex systems.

This paper is based on “Natural Algorithms,” which was published in the *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2009, and subsequent work.

1.3. What is complexity?

Such is the appeal of the word “complexity” that it comes in at least four distinct flavors.

- *Semantic*: What is hard to understand. For 99.99% of mankind, complex means complicated.
- *Epistemological*: What is hard to predict. An independent notion altogether: complex chaotic systems can be simple to understand while complicated mechanisms can be easy to predict.
- *Instrumental*: What is hard to compute, the province of theoretical computer science.
- *Linguistic*: What is hard to describe. Physics has low descriptive complexity—that is part of its magic.^a By contrast, merely specifying a natural algorithm may require an arbitrarily large number of variables to model the diversity present in the system. To capture this type of complexity is a distinctive feature of natural algorithms.

1.4. Beyond simulation

Decades of work in programming languages have produced an advanced theory of abstraction. Ongoing work on reactive systems is attempting to transfer some of this technology to biology⁹. Building on years of progress in automata theory, temporal logic, and process algebra, the goal has been to build a modeling framework for biological systems that integrate the concepts of concurrency, interaction, refinement, encapsulation, modularity, stochasticity, causality, and so on. With the right specifications in place, the hope is that established programming language tools, such as type theory, model checking, abstract interpretation, and the π -calculus can aid in verifying temporal properties of biosystems. The idea is to reach beyond numerical simulation to analyze the structure and interactions of biological systems.

Such an approach, however, can only be as powerful as the theory of natural algorithms behind it. To illustrate this point, consider classifying all possible sequences $\mathbf{x}$, $P\mathbf{x}$, $P^2\mathbf{x}$, $P^3\mathbf{x}$, and so on, where $\mathbf{x}$ is a vector and P is a fixed stochastic matrix. Simulation, machine learning, and verification techniques can help, but no genuine understanding of the process can be achieved without Perron–Frobenius theory. Likewise, natural algorithms need not only computers but also a theory. The novelty of the theory will be its reliance on algorithmic proofs—more on this below.

1.5. Algorithms from nature

If living processes are powered by the “software” of nature, then natural selection is the ultimate code optimizer. With time and numbers on their side—billions of years and 10^{30} living specimens—bacteria have had ample opportunity to perfect their natural algorithm. No wonder computer scientists are turning to biology for algorithmic insight: neural nets and DNA computing, of course, but also ant colony optimization³, shortest path algorithms in slime molds²;

^a Descriptive complexity is an established subfield of theoretical computer science and logic. Our usage is a little different, but not different enough to warrant a change of terminology.

maximal independent sets in fly brain development¹, and so on. Consensus, synchronization, and fault tolerance are concepts central to both biology and distributed computing^{15,17}. The trade of ideas promises to be flowing both ways. This article focuses on the outbound direction: how algorithmic ideas can enrich our understanding of nature.

2. INFLUENCE SYSTEMS

A bad, fanciful script will make a good stage-setter. One fateful morning, you stumble out of bed and into your kitchen only to discover, crawling on the floor, a swarm of insects milling around. Soon your dismay turns to curiosity, as you watch the critters engage in a peculiar choreography. Each insect seems to be choosing a set of neighbors (living or inert) and move either toward or away from them. From what you can tell, the ants pick the five closest termites; the termites select the nearest soil pellets; the ladybugs pick the two ants closest to the powdered sugar that is not in the vicinity of any termite; and so on. Each insect seems equipped with its own selection procedure to decide how to pick neighbors based on their species and the local environment. Once the selection is made, each agent invokes a second procedure, this time to move to a location determined entirely by the identities and positions of its neighbors. To model this type of multiagent dynamics, we have introduced *influence systems*⁶: the model is new, but only new to the extent that it unifies a wide variety of well-studied domain-dependent systems. Think of influence systems as a brand of networks that perpetually rewire themselves endogenously.

2.1. Definition and examples

An *influence system*, is specified by two functions f and $\mathcal{G}$: it is a discrete-time dynamical system,^b $\mathbf{x} \mapsto f(\mathbf{x})$ in $(\mathbb{R}^d)^n$, where n is the number of agents, d is the dimension of the ambient space ($d = 2$ in the example above), and each “coordinate” x_i of the state $\mathbf{x} = (x_1, \dots, x_n) \in (\mathbb{R}^d)^n$ is a d -tuple encoding the location of *agent* i in $\mathbb{R}^d$. With any state $\mathbf{x}$ comes a directed “communication” graph, $\mathcal{G}(\mathbf{x})$, with one node per agent. Each coordinate function f_i of the map $f = (f_1, \dots, f_n)$ takes as input the neighbors of agent i in $\mathcal{G}(\mathbf{x})$, together with their locations, and outputs the new location $f_i(\mathbf{x})$ of agent i in $\mathbb{R}^d$. The (action) function f and (communication) function $\mathcal{G}$ are evaluated by deterministic or randomized algorithms. An influence system is called *diffusive* if f keeps each agent within the convex hull of its neighbors. Diffusive systems never escape to infinity and always make consensus ($x_1 = \dots = x_n$) a fixed point. The system is said to be *bidirectional* if the communication graph always remains undirected.

While f and $\mathcal{G}$ can be arbitrary functions, the philosophy behind influence systems is to keep f simple so that emergent phenomena can be attributed not so much to the power of individual agents, but to the flow of information across the communication network $\mathcal{G}(\mathbf{x})$. By distinguishing $\mathcal{G}$ from f , the model also separates the syntactic (who talks to whom?)

^b A dynamical system generates an orbit by starting at a point $\mathbf{x}$ and iterating the function f to produce $f(\mathbf{x})$, $f^2(\mathbf{x})$, and so on. The goal is to understand the geometry of these orbits. We write the phase space $\mathbb{R}^{dn}$ as $(\mathbb{R}^d)^n$ to emphasize that the action is on the n agents embedded in d -space.

from the semantic (who does what?). It is no surprise then to see recursive graph algorithms play a central role and provide a dynamic version of *renormalization*, a technique used in quantum mechanics and statistical physics.

- *Bounded-confidence systems*: In this popular model of social dynamics¹⁰, $d = 1$ and x_i is a real number denoting the “opinion” of agent i . That agent is linked to j if and only if $|x_i - x_j| \leq 1$. The action function f instructs each agent to move to the mass center of their neighbors. This is the prototypical example of a bidirectional diffusive influence system.
- *Sync*: An instance of Kuramoto synchronization, this diffusive influence system links each of n fireflies to the other fireflies whose flashes it can spot. Every critter has its own flashing oscillator, which becomes coupled with those of its neighbors. The function f specifies how the fireflies adjust their flashings in reaction to the graph-induced couplings^{3, 23}. The model has been applied to Huygens’s pendulum clocks as well as to circadian neurons, chirping crickets, microwave oscillators, yeast cell suspensions, and pacemaker cells in the heart²⁵.
- *Swarming*: The agents may be fish or birds, with states encoding positions and velocities (for rigid 3D animal models, $d = 9$). The communication graph links every agent to some of its nearest neighbors. The function f instructs each agent to align its velocity with those of its neighbors, to move toward their center of gravity, and to fly away from its perilously close neighbors^{21, 24}.
- *Chemotaxis*: Some organisms can sense food gradients and direct their motion accordingly. In the case of bacterial chemotaxis, the stimuli are so weak that the organisms are reduced to performing a random walk with a drift toward higher food concentrations. Influence systems can model these processes with the use of both motile and inert agents.^c Chemotaxis is usually treated as an asocial process (single agents interacting only with the environment). It has been observed, however, that schooling behavior can facilitate gradient climbing for fish, a case where living in groups enhances foraging ability¹⁹.

Other examples of influence systems include the Ising model, neural nets, Bayesian social learning, protein–protein interaction networks, population dynamics, and so on.

2.2. How expressive are influence systems?

If agent i is viewed as a computing device, then the d -tuple x_i is its memory. The system is Markovian in that all facts about the past with bearing on the future are encoded in x . The communication graph allows the function f to be local if so desired. The procedure $\mathcal{G}$ itself might be local even when appearance suggests otherwise: for example, to identify

your nearest neighbor in a crowd entails only local computation, although, mathematically, the function requires knowledge about everyone. The ability to encode different action/communication rules for each agent is what drives up the *descriptive complexity* of the system. In return, one can produce great behavioral richness even in the presence of severe computational restrictions.

- Learning, competition, hierarchy*: Agents can implement game-theoretic strategies in competitive environments (e.g., pursuit-evasion games) and learn to cooperate in groups (e.g., quorum sensing). They can self-improve, elect leaders, and stratify into dominance hierarchies.
- Coarse-graining*: Flocks are clusters of birds that maintain a certain amount of communicative cohesion over a period of time. We can view them as “super-agents” and seek the rules governing interaction among them. Iterating in this fashion can set the stage for dynamic renormalization in a time-changing analog of the renormalization group of statistical mechanics (see Section 6).
- Asynchrony and uncertainty*: In the presence of delayed or asynchronous communication, agents can use their memory to implement a clock for the purpose of time stamping. Influence systems can also model uncertainty by limiting agents’ access to approximations of their neighbors’ states.

A few words about our agent-based approach. Consider the diffusion of pollen particles suspended in water. A *Eulerian* approach to this process seeks a differential equation for the concentration $c(x, t)$ of particles at any point x and time t . There are no agents, just density functions evolving over time¹⁸. An alternative approach, called *Lagrangian*, would track the movement of all the individual particles and water molecules by appealing to Newton’s laws. Given the sheer number of agents, this line of attack crashes against a wall of intractability. One way around it is to pick a single imaginary self-propelled agent and have it jiggle about randomly in a Brownian motion. This agent models a typical pollen particle—typical in the “ergodic” sense that its time evolution mimics the space distribution of countless particles caught on film in a snapshot. Scaling plays a key role: our pollen particles indeed can be observed only on a timescale far larger than the molecular bumps causing the jiggling. Luckily, Brownian motion is scale-free, meaning that it can be observed at any scale. As we shall see in Section 6, the ability to express a dynamical process at different scales is an important feature of influence systems.

The strength of the Eulerian approach is its privileged access to an advanced theory of calculus. Its weakness lies in two commitments: global behavior is implied by *infinitesimal* changes; and every point is subject to *identical* laws. While largely true in physics, these assumptions break down in the living world, where diversity, heterogeneity, and autonomy prevail. Alas, the Lagrangian answer, agent-based modeling, itself suffers from a serious handicap: the lack of a theory of natural algorithms.

^c Some living systems (e.g., ants and termites) exchange information by *stigmergy*: instead of communicating directly with signals, they leave traces such as pheromones in the environment, which others then use as cues to coordinate their collective work. Although lacking autonomy, inert components can still be modeled as agents in an influence system.

2.3. What could go wrong?

There are two arguments against the feasibility of a domain-independent theory of natural algorithms. One is an intrinsic lack of structure. But the same can be said of graphs, a subject whose ties to algebra, geometry, and topology have met with resounding success, including the emergence of universality. A more serious objection is that, before we can analyze a natural algorithm, we must be able to specify it. The bottom-up, reductionist approach, which consists of specifying the functionality and communicability of each agent by hand (as is commonly done in swarming models), might not always work. Sometimes, only a top-down, phenomenological approach will get us started on the right track. We have seen this before: the laws of thermodynamics were not derived as abstractions of Newton's laws—though that is what they are—but as rules governing observable macrostates (P , V , T , and all that). Likewise, a good flocking model might want to toss aside anthropomorphic rules (“stay aligned,” “don’t stray,” etc.) and choose optimized statistical rules inferred experimentally. The specs of the natural algorithm would then be derived algorithmically. This is why we have placed virtually no restriction on the communication function $\mathcal{G}$ in our present analysis of diffusive influence systems.

2.4. Outline

We open the discussion in Section 3 with the *total s-energy*, a key analytical device used throughout. We turn to bird flocking in Section 4, which constitutes the archetypical nondiffusive influence system. We take on the diffusive case in Section 5, with the notion of *algorithmic calculus* discussed in Section 6.^d General diffusive influence systems are Turing-complete, yet the mildest perturbation creates periodic behavior. This is disconcerting. Influence systems model how people change opinions over time as a result of human interaction and knowledge acquisition. Instead of ascending their way toward enlightenment, however, people are doomed to recycle the same opinions in perpetuity—there is a deep philosophical insight there, somewhere.

3. THE -ENERGY

This section builds the tools needed for the bidirectional case and can be read separately from the rest. Let $(P_t)_{t \geq 0}$ be an infinite sequence of n -by- n stochastic matrices; stochastic means that the entries are nonnegative and the rows sum up to 1. Leaving aside influence systems for a moment, we make no assumption about the sequence, not even that it is produced endogenously. We ask a simple question about matrix sequences: under what conditions does $P_{<t} := P_{t-1} \dots P_0$ converge as $t \rightarrow \infty$? Certainly not if

$$P_t = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The problem here is lack of *self-confidence*: the two agents in the system do not trust themselves and follow their

neighbors blindly. So let us assume that the diagonal entries of P_t are positive. Alas, this still does not do the trick: the matrices

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2/3 & 1/3 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2/3 & 0 & 1/3 \end{pmatrix}$$

grant the agents self-confidence, yet composing them in alternation exchanges the vectors $(0, 1, 1/4)$ and $(0, 1, 3/4)$ endlessly. The oscillation is caused by the lack of *bidirectionality*: for example, the recurring link from agent 3 to agent 1 is never reciprocated. The fix is to require the graphs to be undirected or, equivalently, the matrices to be type-symmetric,^e which instills mutual confidence among the agents. With both self-confidence and bidirectionality in place, the sequence $P_{<t}$ *always* converges^{12, 14, 16}. (Interestingly, this is *not* true of forward products $P_0 \dots P_t$ in general.) With nothing keeping $(P_t)_{t \geq 0}$ from “stalling” by featuring arbitrarily long repeats of the identity matrix, bounding the convergence rate is obviously impossible. Yet an analytical device, the *total s-energy*⁵, allows us to do just that for bidirectional diffusive influence systems. The trick is to show that they cannot stall too long without dying off.

3.1. Preliminaries

Fix a small $\rho > 0$ and let $(P_t)_{t \geq 0}$ be a sequence of stochastic matrices such that $\rho \leq (P_t)_{ii} \leq 1 - \rho$ and $(P_t)_{ij} > 0 \Rightarrow (P_t)_{ji} > 0$. Let G_t be the (undirected) graph whose edges are the positive entries in P_t . With $\mathbf{x}(t+1) = P_t \mathbf{x}(t)$ and $\mathbf{x}(0) = \mathbf{x} \in [0, 1]^n$, the *total s-energy* is defined as

$$E(s) = \sum_{t \geq 0} \sum_{(i,j) \in G_t} |x_i(t) - x_j(t)|^s. \quad (1)$$

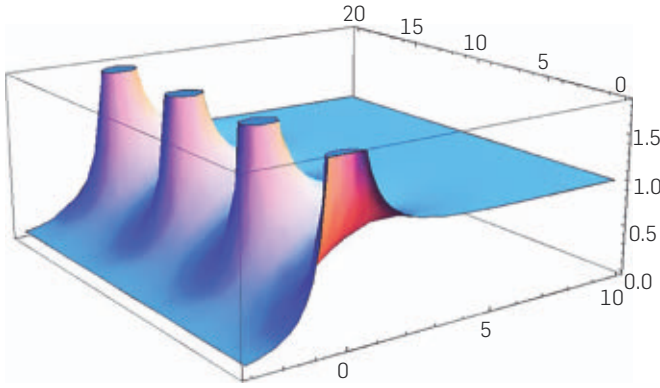
We use the *s-energy* in much the same way one uses Chernoff's inequalities to bound the tails of product distributions. Being a generalized Dirichlet series, the *s-energy* can be inverted and constitutes a lossless encoding of the edge lengths. Why this unusual choice? Because, as with the most famous Dirichlet series of all, the Riemann zeta function $\sum n^{-s}$, the system's underlying structure is multiplicative: indeed, just as n is a product of primes, $x_i(t) - x_j(t)$ is a product of the form $v^t P_{t-1} \dots P_0 \mathbf{x}$. Let $E_n(s)$ denote the maximum value of $E(s)$ over all $\mathbf{x} \in [0, 1]^n$. One should expect the function $E_n(s)$ to encode all sorts of symmetries. This is visible in the case $n = 2$ by observing that it can be continued meromorphically in the whole complex plane (Figure 1).

The sequence formed by $(P_t)_{t \geq 0}$ is called *reversible* if G_t is connected and there is a probability distribution $(\pi_1, \dots, \pi_n)$ such that $\pi_i(P_t)_{ij} = \pi_j(P_t)_{ji}$ for any t ; see detailed definition in Chazelle⁵. This gives us a way to weight the agents so that their mass center never moves. The notion generalizes the concept of reversible Markov chains, with which it shares some of the benefits, including faster convergence to equilibrium.

^d Unless noted otherwise, the results discussed are from Chazelle⁵ for Section 3; Chazelle⁴ for Section 4; Chazelle⁶ for sections 5 and 6.

^e The zero-entries of a *type-symmetric* matrix and its transpose occur at the same locations.

Figure 1. The analytic continuation of $|\mathbb{E}_2(s)|$.



3.2. Bounds

The s -energy measures the total length of all the edges for $s = 1$ and counts their number for $s = 0$; the latter is usually infinite, so it is sensible to ask how big $E_n(s)$ can be for $0 < s \leq 1$. On the lower bound front, we have $E_n(1) = \Omega(1/\rho)^{\ln(2)}$ and $E_n(s) = s^{1-n}(1/\rho)^{\Omega(n)}$, for any n large enough, $s \leq s_0$, and any fixed positive $s_0 < 1$. Of course, the s -energy is useful mostly for its upper bounds⁵:

$$E_n(s) \leq \begin{cases} (1/\rho)^{\Omega(n)} & \text{for } s = 1; \\ s^{1-n} (1/\rho)^{n^2 + O(1)} & \text{for } 0 < s < 1. \end{cases} \quad (2)$$

For reversible sequences and any $0 < s \leq 1$.

$$n^{-2} E_n(s) \leq E_n^D(s) \leq \frac{2n}{s} \left(\frac{2n}{\rho} \right)^{s/2+1}, \quad (3)$$

where $E_n^D(s) = \sum_{t \geq 0} \text{diam}^s \{x_1(t), \dots, x_n(t)\}$. This is essentially optimal. Fix an arbitrarily small $\varepsilon > 0$. A step t is called *trivial* if $|x_i(t) - x_j(t)| < \varepsilon$ for each $(i, j) \in G_t$. The maximum number C_ε of nontrivial steps is bounded by $\varepsilon^{-s} E_n(s)$; hence,

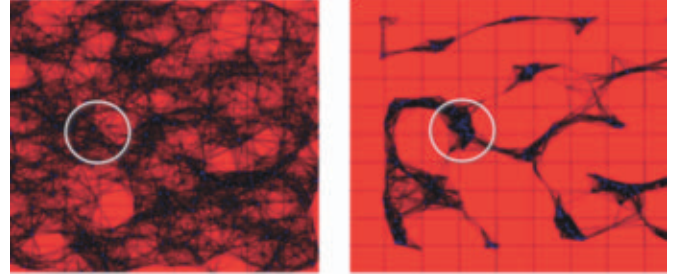
$$C_\varepsilon(n) \leq \min \left\{ \left(\frac{1}{\varepsilon} \right)^{\Omega(n)}, \left(\log \frac{1}{\varepsilon} \right)^{n-1} \left(\frac{1}{\rho} \right)^{n^2 + O(1)} \right\} \quad (4)$$

which is optimal if ε is not too small. Convergence in the reversible case is polynomial: if $\varepsilon < \rho/n$, then $\|\mathbf{x}(t) - \pi^T \mathbf{x}\|_2 \leq \varepsilon$, for $t = O(\rho^{-1} n^2 \log \varepsilon)$. This bound is optimal. In particular, we can specialize it to the case of random walks in undirected graphs and retrieve the usual mixing times.

We briefly mention two examples of diffusive influence systems for which the s -energy readily yields bounds on the convergence time⁵. *HK systems* track opinion polarization in a population¹⁰: in the bounded-confidence version (mentioned earlier), the agents consist of n points in $\mathbb{R}^d$. At each step, each agent moves to the mass center of the agents within distance 1 (Figure 2).

Truth-seeking systems assume a “cognitive division of labor”¹¹. We fix one agent, the truth, and keep the $n - 1$ others mobile. A “truth seeker” is a mobile agent that is joined to the truth in every G_t . All the other mobile agents are “ignorant,” meaning that they never join to the truth through an edge, although they might indirectly communicate with it

Figure 2. Randomly placed agents evolve to form shapes of dimension 2, then 1, and finally 0.



along a path. Any two mobile agents are joined in G_t whenever their distance is less than 1.

3.3. Why the -energy?

Let $\text{conv}P$ denote the convex hull of the points formed by the rows of the matrix P . We have the “Russian doll” nesting structure (Figure 3):

$$\text{conv}P_{\ell_t} \subseteq \dots \subseteq \text{conv}P_1 \subseteq \text{conv}P_0 \subseteq \mathbb{R}^n.$$

The literature on stochastic matrices features a variety of *coefficients of ergodicity* to help us measure how quickly the Russian dolls deflate: eigenvalues, joint spectral radius, width, diameter, volume, and so on²². This is how the convergence of products of stochastic matrices, which includes the whole subject of Markov chain mixing, is established. By seeking progress at each step, however, these methods cannot cope with stalling. The total s -energy gets around this by factoring time out (as a Fourier transform would with a time series) and producing a global deflation measure integrated over the whole time horizon.

The s -energy is controlled by a single parameter s , which we can adjust at will to get the most out of the inequality $C_\varepsilon \leq \varepsilon^{-s} E(s)$, typically choosing s so that $(dE/ds)|_s = E \ln \varepsilon$. We sketch the proof of (2), beginning with the case $s < 1$. The argument relies on an importance device: the *flow tracker*. Think of it as breadth-first search in a dynamic graph. A little imagery will help. Pick agent 1 and dip it in water, keeping all the other agents dry. Whenever an edge of G_t links a dry agent to a wet one, the dry one gets wet. As soon as all the agents become wet (if ever), dry them all except agent 1; repeat.

$$\text{Flow tracker} \quad (5)$$

- [1] $t_0 \leftarrow 0$.
- [2] Repeat forever:
 - [2.1] $W_{t_0} \leftarrow \{1\}$.
 - [2.2] For $t = t_0, t_0 + 1, \dots, \infty$:
 $W_{t+1} \leftarrow W_t \cup \{i \mid \exists (i, j) \in G_t \text{ \& } j \in W_t\}$.
 - [2.3] If $|W_\infty| = n$ then $t_0 \leftarrow \min\{t > t_0 : |W_t| = n\}$ else stop.

Let W_t denote the set of wet agents at time t , which always includes agent 1. The assignments of t_0 in step [2.3] divide the timeline into *epochs*, time intervals during which either all agents become wet or, failing that, the flow tracker

comes to a halt (breaking out of the repeat loop at “stop”). Take the first epoch: it is itself divided into subintervals by the *coupling times* $t_1 < \dots < t_\ell$ at which the set of wet agents grows: $W_{t_k} \subset W_{t_{k+1}}$. If $\|W_\ell\|$ denotes the length of the smallest interval enclosing W_ℓ , it can be shown by induction that $\|W_{t_{k+1}}\| \leq 1 - \rho^k$. It then follows that $E_1(s) = 0$ and, for $n \geq 2$,

$$E_n(s) \leq 2nE_{n-1}(s) + (1 - \rho^n)^s E_n(s) + n^3,$$

which implies (2) for $s < 1$.

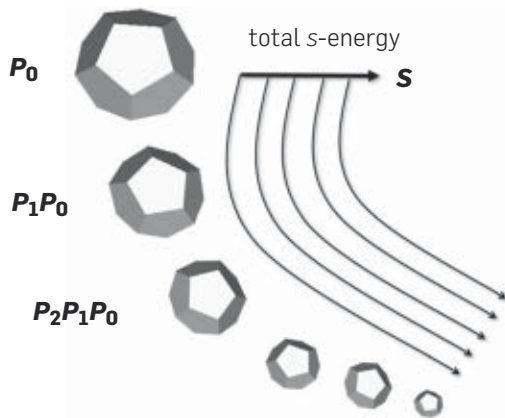
The case $s = 1$ features a fundamental concept in the study of natural algorithms, that of an *algorithmic proof*. Think of the agents as car drivers: the 1-energy can then be shown to measure the total mileage. Fill the gas tank of each car with an initial amount determined by some formula (whose details are immaterial for our purposes). Since we do not know ahead of time how much gas any given car will need, we set up a gas trading mechanism: a refueling tanker hovers over the cars, ready to provide (or take) gas to (or from) any car that needs it. The needs come in two ways: first, a car needs gas to move; second, the trading mechanism specifies how much gas any car must have at any time, so any excess supply must be handed over to the refueling tanker. Any car in need of gas (for driving or simply complying with the rules) is free to help itself from the fuel tanker. The trick is to show that the system can run forever without the fuel tanker ever running out of gas and being unable to meet the drivers' needs.

Dynamicists might think of this as a kind of distributed Lyapunov function. This would be missing the point. Algorithmic proofs of the type found in the amortized analysis of algorithms—often expressed, as above, via the trading rules of a virtual economy—are meant to cope with the sort of descriptive complexity typically absent from low-dimensional dynamics. The benefits come from the richer analytical language of algorithmic proofs: indeed, it is all about language!

4. BIRD FLOCKING

We briefly discuss a classic instance of a nondiffusive influence system, bird flocking, and report the results from Chazelle⁴. The alignment model we use^{8, 13, 24} is a

Figure 3. The deflating matrix polytope.



trimmed-down version of Reynolds's original model²¹. In this influence system, $d = 6$ and each bird i is specified by its position z_i and velocity v_i . The undirected communication graph $\mathcal{G}(\mathbf{x})$ joins any two birds within a certain fixed distance of each other (Figure 4). The birds in any connected component form a *flock*. Reordering the coordinates of the $6n$ -dimensional state vector $\mathbf{x}$ as $(\mathbf{z}, \mathbf{v})$, we specify the dynamics as

$$\mathbf{x} \xrightarrow{f} (P(\mathbf{x}) \otimes \mathbf{I}_3) \mathbf{x} := \left\{ \begin{pmatrix} \mathbf{I}_n & \mathbf{I}_n \\ 0 & Q(\mathbf{x}) \end{pmatrix} \otimes \mathbf{I}_3 \right\} \mathbf{x},$$

where $Q(\mathbf{x})$ is the n -by- n “velocity” stochastic matrix of a (lazy) random walk on the graph $\mathcal{G}(\mathbf{x})$ and $\otimes$ is the Kronecker product that distributes the action along each coordinate axis. The matrix P is $2n$ -by- $2n$, so each entry is multiplied not by a single coordinate in $\mathbf{x}$ but by a 3-tuple; in other words, $P(x)$ acts not on $\mathbb{R}^{2n}$ but on $(\mathbb{R}^3)^{2n}$. Although the velocities can be inferred from the positions, they need to be included in the phase space to keep the system Markovian.

The system always converges in the following sense: after an initial period when the behavior can be fairly arbitrary, the birds aggregate in flocks which, from that point on, can only merge together. If we wait long enough, the flocks will eventually stabilize. The communication graph will remain forever fixed and the flocks will each move at a constant speed and never meet again. The fragmentation period is at most exponential in the bit length of the input. The convergence of birds' velocities requires the use of the total s -energy and various geometric considerations. One of them is to show that some *virtual bird* must necessarily fly almost along a straight line after a while. This addresses the issue of whether all birds can keep flying in spirals.

What is a virtual bird? Imagine the presence of one baton in the system. At any given time, a single bird holds the baton and may pass it on to any of its neighbors in the communication graph (Figure 5). Whoever holds the baton is the virtual bird at that instant (virtual because its identity keeps changing). Is there a baton-passing protocol that will keep the virtual bird flying in (almost) a straight line? (A question whose answer can only be an algorithmic proof.) This is key to determining whether two flocks flying away from each

Figure 4. The bird at the center of the circle is influenced by its two neighbors in it.

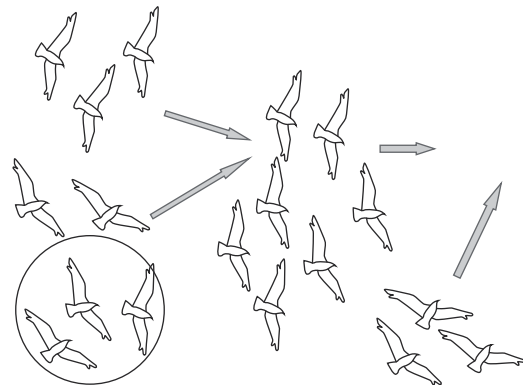


Figure 5. The virtual bird can hold on to the baton or pass it to any neighbor.

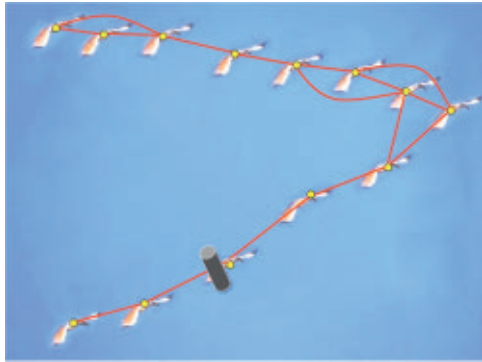


Figure 6. The flight net and path integrals in $\mathbb{R}^4$.



other can be brought together by other flocks. The design of a protocol involves the geometric analysis of the *flight net* (Figure 6), which is the unfolding of all neighboring relations in four-dimensional spacetime. This is a tree-like geometric object in $\mathbb{R}^4$ with local convexity properties, which encodes the exponentially large number of influences among birds. The position of each bird can be expressed by a path integral in that space. Examining these integrals (actually sums) and the geometry that produces them allows us to answer the question above⁴.

While flocks cease to fragment reasonably rapidly (thus ensuring quick physical convergence), it might take very long for them to stop merging. How long? A tower-of-twos of height logarithmic in the number of birds. Surprisingly, this result is tight! To prove that, we regard the set of birds as forming a computer and we ask a “busy-beaver” type of question: What is the smallest nonzero angle between any two stationary velocities? The term “stationary” refers to the fact that each flock is a coupled oscillator with constant stationary velocity (the lowest mode of its spectrum). These velocity vectors form angles between them. How small can they get short of 0? (Zero angles correspond to flocks flying in parallel.) To answer this question requires looking at the flocks of birds as a circuit whose gates, enacting flock merges, produce a redistribution of the energy among the modes called a *spectral shift*. It is remarkable that the busy-beaver function of this exotic computing device can

be pinned down almost exactly: the logarithmic height of the tower-of-twos is known up to a factor of 4.

5. DIFFUSIVE INFLUENCE SYSTEMS

We set $f(\mathbf{x}) = (P(\mathbf{x}) \otimes \mathbf{I}_d) \mathbf{x} \in (\mathbb{R}^d)^n$, where $P(\mathbf{x})$ is a stochastic matrix whose positive entries correspond to the edges of $\mathcal{G}(\mathbf{x})$ and are rationals assumed larger than some arbitrarily small $\rho > 0$. We grant the agents a measure of self-confidence by adding a self-loop to each node of $\mathcal{G}(\mathbf{x})$. Agent i computes the i -th row of $P(\mathbf{x})$ by means of its own algebraic decision tree; that is, on the basis of the signs of a finite number of dn -variate polynomials evaluated at the coordinates of $\mathbf{x}$. This high level of generality allows $\mathcal{G}(\mathbf{x})$ to be specified by any first-order sentence over the reals.[†] Note how the descriptive complexity resides in the communication algorithm $\mathcal{G}$, which can be arbitrarily expressive: the action f is confined to diffusion (with different weights for each agent if so desired).

By taking tensor powers (no details needed here), we can linearize the system and reduce the dimension to $d = 1$, so $f(\mathbf{x}) = P(\mathbf{x})\mathbf{x}$, where $P(\mathbf{x}) = P_c$ for any $\mathbf{x} \in c$, and c is an *atom* (open n -cell) of an arrangement of hyperplanes in $\mathbb{R}^n$, called the *switching partition SP* (Figure 7). We assume self-confidence but not mutual confidence, that is, positive diagonal entries but not necessarily bidirectionality.

In spite of having no positive Lyapunov exponents, diffusive systems can be chaotic and even Turing-complete. Perturbations wipe out their computational power, however, by making them attracted to periodic orbits. Such systems, in other words, are clocks in disguise.[‡] This dichotomy requires a subtle bifurcation analysis, which we sketch in the next section.

THEOREM 1:⁶ *Given any initial state, the orbit of a diffusive influence system is attracted exponentially fast to a limit cycle almost surely under an arbitrarily small perturbation. The period and preperiod are bounded by a polynomial in the reciprocal of the failure probability. In the bidirectional case, the system is attracted to a fixed point in time $n^{O(n)}|\log \varepsilon|$, where n is the number of agents and ε is the distance to the fixed point.*

The number of limit cycles is infinite, but if we measure distinctness the right way (i.e., by factoring out foliations), there are actually only a finite number of them.^h The critical region of parameter space is where chaos, strange attractors, and Turing completeness reside. It is still very mysterious.ⁱ This does not mean that it is difficult to identify at least some of the critical points. Here is a simple

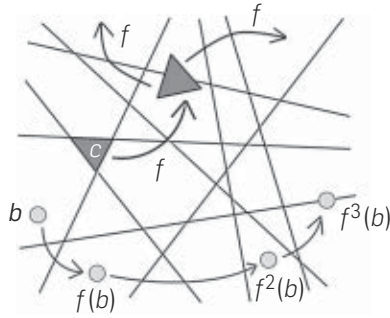
[†] This is the language of geometry and algebra over the reals with statements specified by any number of quantifiers and polynomial (in)equalities. It was shown to be decidable by Tarski and amenable to quantifier elimination and algebraic cell decomposition by Collins⁷.

[‡] To perturb the system means randomly perturbing the switching partition and making exceptions for infinitesimal and indefinitely disappearing edges. The conditions are easy to enforce and, in one form or another, required. Note that we do not perturb the matrices or the initial states.

^h A limit cycle is an attracting periodic orbit, for example, $\{-1, 1\}$ where $x(t) = (-1)^t + 2^{-t} x(t-1)$ and $x(0) \in \mathbb{R}$.

ⁱ I only know that the critical region has measure zero and looks like a Cantor set.

Figure 7. The atom c of the SP maps via f to a cell intersecting two atoms.



example of a chaotic diffusive influence system with $n = 4$ and $d = 1$: the first two agents stay on opposite sides of the origin; at any time, the one further from the origin moves to their midpoint, that is,

$$(x_1, x_2) \xrightarrow{f} \frac{1}{2} \begin{cases} (2x_1, x_1 + x_2) & \text{if } x_1 + x_2 \geq 0 \\ (x_1 + x_2, 2x_2) & \text{else.} \end{cases}$$

Agent 3 is fixed at $x_3 = 1$. Agent 4 moves midway toward agent 1 if the latter is moving and midway toward agent 3 if agent 2 is the one moving. To see why the system has positive topological entropy (the usual sign of chaos), it suffices to consider the infinite bit string $s_0 s_1 s_2 \dots$, where $s_t = 0/1$ depends on whether agent 1 or 2 is moving at time t . If agent 2 is initialized at $x_2 = 1$ and agent 1 anywhere in $(-1, 0)$, the string matches the binary expansion of $1/(1 - x_1)$; in other words, predicting the t -th step requires knowing the initial placement of agent 1 with an error exponentially small in t .

5.1. Energy vs. entropy

As in the Ising model, the system mediates a clash between two opposing “forces”: one, caused by the map’s discontinuities, is “entropic” and leads to chaos; the other one, related to the Lyapunov exponents, is energetic and pulls the system toward an attracting set within which the dynamics is periodic. The goal is to show that, outside a vanishingly small “critical” region in parameter space, entropy always loses. What does it mean? If, unlike in Figure 7, the iterated image of any ball b never intersected the SP hyperplanes, it would merrily bounce around until eventually periodicity kicked in. In the figure, $f^3(b)$ does not oblige and splits into two smaller bodies. Both will bounce around until possibly splitting again and so on. If this branching process gets out of control, chaos will ensue. To squelch this entropic process and induce periodicity, we have the stochasticity of the matrices on our side: it causes the ball b to shrink and dissipate energy. Unlike the classical Ising model, however, the system has a single phase outside the critical region.

Entropy against energy: which one will win? For entropy to lose out, the ball b must avoid splitting up too frequently. This can be expressed by an (infinite) system of linear inequalities. Feasibility hinges on a type of *matrix rigidity* question: in this case, given a certain matrix, how many rows

must be removed before we can express the first column as a linear combination of the others? Periodicity requires that this number be high. The matrix in question is extracted from the system’s stochastic matrices and the SP equations, hence is highly structured.

Our sketchy explanation skipped over the main source of difficulty. The ball b in fact does not shrink in all directions. Take $(1, \dots, 1)^T$: it is a principal right-eigenvector for the eigenvalue 1, so we would not see much contraction in that direction. Worse, the noncontracting principal eigenspace can have arbitrarily high dimension d_c . To prove that the critical region has measure zero requires a firm analytical grip on d_c . At time t , the dimension d_t is equal to the number of essential communicating classes in the Markov chain $P(f^t(\mathbf{x}))$. To keep track of d_c , a number that can change constantly, we extend the flow tracker to dynamic directed graphs.

6. AN ALGORITHMIC CALCULUS

As the system evolves, we are in a position to isolate certain subsystems and treat them recursively. If, during a certain time interval, the communication network consists of two dynamic subgraphs A, B with no directed edges from B to A , then we can break down the system during that period into B and C , where C consists of A together with the contraction of B into a single vertex. This recursive treatment (“dynamic renormalization”) can only be accomplished algorithmically since we do not know the structure of the recursive systems ahead of time—what with edges coming and going all the time. (Observe the difference with the Ising model, where the particles sit on a lattice and the coarse-graining can be structured prior to action.) We mention another difference, this time with classical algorithms. The decision tree of influence systems (called the coding tree) is infinite, so any perturbation has rippling effects that translate into infinitely many conditions; this explains the need to deal with infinite series such as the total s -energy or infinite sets of algebraic conditions as in the matrix rigidity problem mentioned earlier.

By scale invariance and convexity, we may confine the phase space to $[0, 1]^n$. Let $\mathcal{M}_\delta$ denote the union of all the SP hyperplanes shifted by a small δ . It is useful to classify the initial states by how long it takes their orbits to hit $\mathcal{M}_\delta$, if ever. With $f^0 = \mathbf{I}_n$ and $\min \emptyset = \infty$, we define the label $\ell(\mathbf{x})$ of $\mathbf{x} \in [0, 1]^n$ as the minimum integer t such that $f^t(\mathbf{x}) \in \mathcal{M}_\delta$. The point $\mathbf{x}$ is said to *vanish* at time $\ell(\mathbf{x})$ if its label is finite. The points that do not vanish before time t form the set $\mathcal{S}_t$; we have $\mathcal{S}_0 = [0, 1]^n$; and, for $t > 0$,

$$\mathcal{S}_t = \mathcal{S}_0 \setminus \bigcup_{k=0}^{t-1} f^{-k}(\mathcal{M}_\delta).$$

Each of $\mathcal{S}_t$ ’s connected components is specified by a set of strict linear inequalities in $\mathbb{R}^n$, so $\mathcal{S}_t$ is a union of disjoint open n -cells, whose number we denote by $\#\mathcal{S}_t$. Each cell of $\mathcal{S}_{t+1}$ lies within a cell of $\mathcal{S}_t$. The limit set $\mathcal{S}_\infty = \bigcap_{t \geq 0} \mathcal{S}_t$ collects the points that never vanish. We say that the system is *nesting* at t if $\mathcal{S}_t = \mathcal{S}_{t+1}$. The minimum value of t (or ∞) is called the *nesting time* ν of the system. Labels cannot be skipped: if k

is a label, then so is $k - 1$. It follows that the nesting time v is the minimum t such that, for each cell c of S_v , $f^t(c)$ lies within an atom. If c is a cell of S_v , then $f(c)$ intersects at most one cell of S_v and $S_v = S_\infty$. Any nonvanishing orbit is eventually periodic and the sum of its period and preperiod is bounded by $\#S_v$.

We define the directed graph F with one node per cell c of S_v and an edge from (c, c') , where c' is the unique cell of S_v , if it exists, that intersects $f(c)$. The edge (c, c') is labeled by the linear map $f|_c$ defined by the matrix P_a , where a is the unique atom $a \supseteq c$. The graph defines a sofic shift (i.e., a regular language) of the functional kind, meaning that each node has exactly one outgoing edge, possibly a self-loop, so any infinite path leads to a cycle. Periodicity follows immediately. The *trajectory* of a point x is the string $s(x) = c_0 c_1 \dots$ of atoms that its orbit visits: $f^t(x) \in c_t$ for all $0 \leq t < \ell(x)$. It is infinite if and only if x does not vanish, so all infinite trajectories are eventually periodic.

6.1. The coding tree

This infinite rooted tree $\mathcal{T}$ encodes into one geometric object the set of all orbits. (It is the natural-algorithm equivalent of the decision tree of classical algorithms.) It is embedded in $[0, 1]^n \times \mathbb{N}$, with the last dimension representing time. The atoms are redefined as the n -dimensional cells in $[0, 1]^n \setminus \mathcal{M}_\delta$. Each child v of the root is associated with an atom U_v . The *phase tube* (U_v, V_v) of each child v is the “time cylinder” whose cross sections at times 0 and 1 are U_v and $V_v = f(U_v)$, respectively. The tree is built recursively by subdividing V_v into the cells c formed by its intersection with the atoms, and attaching a new child w for each c : we set $V_w = f(c)$ and $U_w = U_v \cap f^{-t_v}(c)$, where t_v is the depth of v (Figure 8). We denote by P_w the matrix of the map’s restriction to c . The phase tube (U_v, V_v) consists of all the cylinders whose cross sections at $t=0, \dots, t_v$ are, respectively, $U_v, f(U_v), \dots, f^{t_v}(U_v) = V_v$. Intuitively, $\mathcal{T}$ divides up the atoms into maximal regions over which the iterated map is linear.

Let $ww'w'' \dots$ denote the upward, t_w -node path from w to the root (but excluding the root). Using the notation $P_{\leq w} = P_w P_{w'} P_{w''} \dots$, we have the identities $V_w = P_{\leq w} U_w$ and $S_k = \bigcup_w \{U_w \mid t_w = k\}$. Labeling each node w by the unique atom that contains the cell c above allows us to interpret any path as a word of atom labels and define the language $L(\mathcal{T})$ of all such words. The coding tree is the system’s Rosetta stone, from which everything of interest can be read. To do that, we need to define a few parameters:

- The *nesting time* $v = v(\mathcal{T})$ is the minimum depth at which any node has at most one nonvanishing child. A node v is *shallow* if $t_v \leq v$.
- The *word-entropy* $h(\mathcal{T})$ captures the growth rate of the language $L(\mathcal{T})$: it is defined as the logarithm of the number of shallow nodes; $\#S_v \leq 2^{h(\mathcal{T})}$.
- The *period* $p(\mathcal{T})$ is the maximum (prime) period of any word in $L(\mathcal{T})$.

6.2. The arborator

We assemble the coding tree by glueing together smaller coding trees defined recursively. We entrust this task to the

Figure 8. A phase tube (U_w, V_w) of length two.

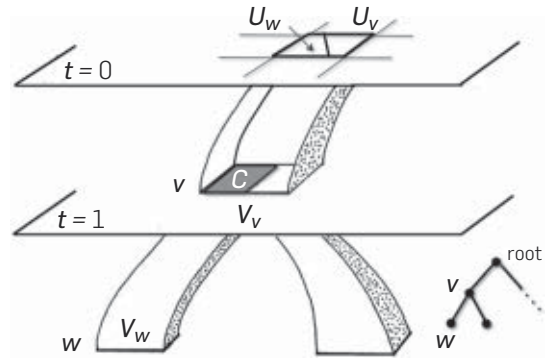
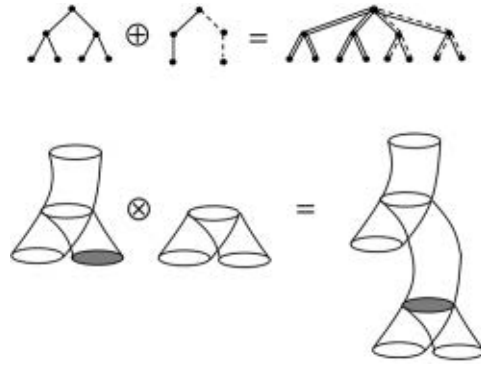


Figure 9. The two tensor operations.

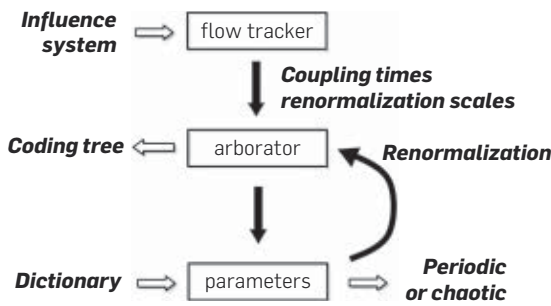


arborator, a recursive algorithm expressed in a language for “lego-like” assembly. The arborator needs two (infinite) sets of parameters to do its job, the *coupling times* and the *renormalization scales*. To produce these numbers, we use an extension of the flow tracker (Section 3) to directed graphs. The arborator relies on a few primitives that we briefly sketch. The direct sum and direct product are tensor-like operations that we use to assemble the coding tree from smaller pieces (Figure 9). We can also compile a *dictionary* to keep track of the tree’s parameters (nesting time, word-entropy, etc.) as we build it up one piece at a time.

Direct sum. The coding tree $\mathcal{T} = \mathcal{T}_1 \oplus \mathcal{T}_2$ models two independent systems of size n_1 and n_2 . The phase space of the direct sum is of dimension $n = n_1 + n_2$. A path $w_0, w_1, \dots$ of $\mathcal{T}$ is a pairing of paths in the constituent trees: the node w_t is of the form (u_t, v_t) , where u_t (respectively, v_t) is a node of $\mathcal{T}_1$ (respectively, $\mathcal{T}_2$) at depth t . The direct sum is commutative and associative; furthermore, $U_w = U_u \times U_v$, $V_w = V_u \times V_v$, and $P_w = P_u \oplus P_v$.

Direct product. Consider two systems S_1 and S_2 , governed by different dynamics yet evolving in the same phase space $[0, 1]^n$. Given an arbitrary region $\Lambda \subset [0, 1]^n$, define the hybrid system S with the dynamics of S_2 over Λ and S_1 elsewhere. Suppose we had complete knowledge of the coding tree $\mathcal{T}_i$ of each S_i ($i = 1, 2$). Could we then combine them in some ways in cut-and-paste style to assemble the coding tree $\mathcal{T}$ of S ? The direct product $\mathcal{T}_1 \otimes \mathcal{T}_2$ provides the answer. The operation is

Figure 10. The algorithmic calculus.



associative but not commutative. It begins by marking certain nodes of $\mathcal{T}_1$ as *absorbed* and pruning the subtrees below. This operation is called *absorption* by analogy with the absorbing states of a Markov chain: any orbit reaching an absorbed leaf comes to a halt, broken only after we reattach a copy of $\mathcal{T}_2$ at that leaf. The copy must be properly cropped.

6.3. Dynamic renormalization

Direct sums model independent subsystems through parallel composition. Direct products model sequential composition. What are the benefits? In pursuit of some form of contractivity, the generalized flow tracker classifies the communication graphs by their connectivity properties and breaks up orbits into sequential segments accordingly (Figure 10). It partitions the set of stochastic matrices into classes and decomposes the coding tree $\mathcal{T}$ into maximal subtrees consisting of nodes v with matrices P_v from the same class. The power of this “renormalization” procedure is that it can be repeated recursively. We classify the communication graphs by their *block-directionality* type: $\mathcal{G}(\mathbf{x})$ is of type $m \rightarrow n - m$ if the agents can be partitioned into A , B ($|A| = m$) so that no B -agent ever links to an A -agent; if in addition, no A -agent links to any B -agent, $\mathcal{G}(\mathbf{x})$ is of type $m \rightarrow n - m$. If we define the renormalization scale $w_k = |W_{t_k+1}| - n + m$ for $k = 1, \dots, \ell - 1$ (where W_t denotes the set of wet nodes), any path of the coding tree can be expressed as

$$\mathcal{T}_{m \rightarrow n-m} \Rightarrow \underbrace{\mathcal{T}_m}_{\|n-m\|_{t_1}} \otimes \left\{ \bigotimes_{k=1}^{\ell-1} \mathcal{T}_{w_k \rightarrow n-w_k} \right\}_{t_{k+1}-t_k-1} \otimes \mathcal{T}_{m \rightarrow n-m}.$$

The subscripts indicate the lengths of the (underlined) renormalized subsystems. Varying the shift δ may change the coding tree, so we extend all the previous definitions to the *global coding tree* $\mathcal{T}^\Delta$ with phase space $[0, 1]^n \times \Delta$, for a tiny interval Δ centered at the origin. We have all the elements in place for the algorithmic proof of Theorem 1 to proceed: see Chazelle⁶ for details.

Acknowledgments

I wish to thank Andrew Appel, Stan Leibler, Ronitt Rubinfeld, David Walker, and the anonymous referee for helpful comments and suggestions. This work was supported in part by NSF grants CCF-0832797, CCF-0963825, and CCF-1016250. 

References

1. Afek, Y., Alon, N., Barad, O., Hornstein, E., Barkai, N., Bar-Joseph, Z. A biological solution to a fundamental distributed computing problem. *Science* 331 (2011), 183–185.
2. Bonifaci, V., Mehlhorn, K., Varma, G. Physarum can compute shortest paths. In *Proceedings of 23rd Annual ACM-SIAM Symposium on Discrete Algorithms* (2012), 233–240.
3. Camazine, S., Deneubourg, J.L., Franks, N., Sneyd, J., Bonabeau, E., Theraulaz, G. *Self-Organization in Biological Systems*, Princeton University Press, 2001.
4. Chazelle, B. The convergence of bird flocking, arXiv:0905.4241v1, 2009. Prelim. version in *Proceedings of SIAM SODA 2009*, with improvements in *Proceedings of ACM SoCG 2010*.
5. Chazelle, B. The total s-energy of a multiagent system, *SIAM J. Control Optim.* 49 (2011), 1680–1706.
6. Chazelle, B. The dynamics of influence systems, arXiv:1204.3946v2, 2012. Prelim. version in *Proceedings of 53rd FOCS*, 2012.
7. Collins, G. E. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings of 2nd GI Conference on Automata Theory and Formal Languages* (1975), Springer-Verlag, New York, 134–183.
8. Cucker, F., Smale, S. Emergent behavior in flocks. *IEEE Trans. Automatic Control* 52 (2007), 852–862.
9. Fisher, J., Harel, D., Henzinger, T.A. Biology as reactivity. *Commun. ACM* 54 (2011), 72–82.
10. Hegselmann, R., Krause, U. Opinion dynamics and bounded confidence models, analysis, and simulation. *J. Artif. Soc. Soc. Simulat.* 5 (2002), 3.
11. Hegselmann R, Krause U. Truth and cognitive division of labor: first steps towards a computer aided social epistemology. *J. Artif. Soc. Soc. Simulat.* 9 (2006).
12. Hendrickx, J.M., Blondel, V.D. Convergence of different linear and non-linear Vicsek models. In *Proceedings of 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS2006)* (July 2006, Kyoto, Japan), 1229–1240.
13. Jadbabaie, A., Lin, J., Morse, A.S. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Trans. Automatic Control* 48 (2003), 988–1001.
14. Lorenz, J. A stabilization theorem for dynamics of continuous opinions. *Phys. Stat. Mech. Appl.* 355 (2005), 217–223.
15. Lynch, N.A. *Distributed Algorithms*, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1996.
16. Moreau, L. Stability of multiagent systems with time-dependent communication links. *IEEE Trans. Automatic Control* 50 (2005), 169–182.
17. Navlakha, S., Bar-Joseph, Z. Algorithms in nature: the convergence of systems biology and computational thinking. *Mol. Syst. Biol.* 7 (2011), 546.
18. Okubo, A., Levin, S.A. *Diffusion and Ecological Problems*, 2nd edn, Springer, 2002.
19. Parrish, J.K., Hamner, W.M. *Animal Groups in Three Dimensions*, Cambridge University Press, 1997.
20. Prusinkiewicz, P., Lindenmayer, A., Hanan, J.S., Fracchia, F.D. *The Algorithmic Beauty of Plants*, Springer-Verlag, 1996.
21. Reynolds, C.W. Flocks, herds, and schools: a distributed behavioral model. *Comput. Graph.* 21 (1987), 25–34.
22. Seneta, E. *Non-Negative Matrices and Markov Chains*, 2nd edn, Springer, 2006.
23. Strogatz, S.H. From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D* 143 (2000), 1–20.
24. Vicsek, T., Czirók, A., Ben-Jacob, E., Cohen, I., Shochet, O. Novel type of phase transition in a system of self-driven particles. *Phys. Rev. Lett.* 75 (1995), 1226–1229.
25. Winfree, A.T. Biological rhythms and the behavior of populations of coupled oscillators. *J. Theoret. Biol.* 16, 1 (1967), 15–42.

Bernard Chazelle (chazelle@cs.princeton.edu), Department of Computer Science, Princeton University.

interactions

EXPERIENCES | PEOPLE | TECHNOLOGY



interactions magazine is pleased to introduce its new website—interactions.acm.org—designed to capture the influential voice of its print component in covering the fields that envelop the study of people and computers.

The site offers a rich history of the conversations, collaborations, and discoveries from issues past, present, and future.

Check out the current issue, look up a past prototype, or discuss an upcoming trend in the communities of design and human-computer interaction.

Welcome to
interactions.acm.org

FEATURES

BLOGS

FORUMS

DOWNLOADS

Association for
Computing Machinery





THE ACM A. M. TURING AWARD

by the community ♦ from the community ♦ for the community



ACM, Intel, and Google congratulate

JUDEA PEARL

for fundamental contributions to artificial intelligence through the development of a calculus for probabilistic and causal reasoning.



"Dr. Pearl's work provided the original paradigm case for how to do statistical AI. By placing structured knowledge representations at the heart of his work, and emphasizing how these representations enabled efficient inference and learning, he showed the field of AI how to build statistical reasoning systems that were actually telling us something about intelligence, not just statistics."

Limor Fix
Director, University Collaborative Research Group
Intel Labs

For more information see www.intel.com/research.

"Judea Pearl is the most prominent advocate for probabilistic models in artificial intelligence. He developed mathematical tools to tackle complex problems that handle uncertainty. Before Pearl, AI systems had more success in black and white domains like chess. But robotics, self-driving cars, and speech recognition deal with uncertainty. Pearl enabled these applications to flourish, and convinced the field to adopt these techniques."

Alfred Spector
Vice President, Research and Special Initiatives
Google Inc.

For more information, see <http://www.google.com/corporate/index.html> and <http://research.google.com/>.



Google™

Financial support for the ACM A. M. Turing Award is provided by Intel Corporation and Google Inc.

American University College of Arts and Sciences Assistant Professor

The College of Arts and Sciences at **American University (Washington, DC)** invites applications for a full-time, tenure-line, **Assistant Professor** position, beginning in August 2013, in computational neuroscience (broadly defined, including but not limited to neural networks, simulation, image processing, and bio-informatics). The appointee's tenure home and departmental affiliation will depend on his or her research background, but we are most interested in applicants who would have at least a joint affiliation with the Department of Computer Science. Applicants must have a PhD in a relevant discipline. Teaching and post-doctoral experience are preferred. Responsibilities include: teaching and curriculum development; establishing an internationally recognized research program, preferably one that can involve undergraduate research participation; strengthening connections to neurosciences across campus; and service to the appointee's home department and the wider university.

American University has made other recent hires in neuroscience, is presently seeking to fill other positions in Computer Science, and benefits from proximity to other scientific institutions in the Washington area. (For example, NIH is three metro stops from the AU campus.) The College of Arts and Sciences offers a variety of degrees at the undergraduate, masters, and doctoral levels. For more information about our programs, visit <http://www.american.edu/cas/>.

Applicants should submit a cover letter, curriculum vitae, teaching statement, and research statement, and applicants must arrange for three letters of recommendation to be sent directly to the search committee. Materials can be submitted online (highly preferred) at <http://academicjobsonline.org/ajo>, or via email to CompNeuroSearch@american.edu, or in hard copy to Computational Neuroscience Search Committee, Department of Mathematics and Statistics, American University, Washington, DC 20016-8050. Applications received by December 10, 2012 will receive full consideration. American University is an EEO/AA institution, committed to a diverse faculty, staff, and student body. Women and minority candidates are strongly encouraged to apply. American University offers employee benefits to same-sex domestic partners of employees and prohibits discrimination on the basis of sexual orientation/preference and gender identity/expression.

Baylor University Lecturer of Computer Science

The Department of Computer Science seeks a dedicated teacher and program advocate for a lecturer position beginning August, 2013. The ideal candidate will have a master's degree or Ph.D. in

Computer Science or a related area, a commitment to undergraduate education, effective communication and organization skills, and industry/academic experience in game development, especially with graphics and/or engine development. For position details and application information please visit: <http://www.baylor.edu/hr/index.php?id=81302>

Baylor, the world's largest Baptist university, holds a Carnegie classification as a "high-research" institution. 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 deepening its distinctive Christian mission as described in *Pro Futuris* (<http://www.baylor.edu/profuturis/>).

Baylor is a Baptist university affiliated with the Baptist General Convention of Texas. As an AA/EEO employer, Baylor encourages minorities, women, veterans, and persons with disabilities to apply.

Baylor University Assistant, Associate or Full Professor of Computer Science

The Department of Computer Science seeks a productive scholar and dedicated teacher for a tenured or tenure-track position beginning August, 2013. The ideal candidate will hold a terminal degree in Computer Science or a closely related field and 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 and 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>

Baylor, the world's largest Baptist university, holds a Carnegie classification as a "high-research" institution. 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 deepening its distinctive Christian mission as described in *Pro Futuris* (<http://www.baylor.edu/profuturis/>).

Baylor is a Baptist university affiliated with the Baptist General Convention of Texas. As an AA/EEO employer, Baylor encourages minorities, women, veterans, and persons with disabilities to apply.

Bowling Green State University THREE Tenure-track positions in CS

We are seeking to fill three tenure-track Assistant Professor positions in Computer Science to teach a variety of courses at the undergraduate and graduate levels and to be productive in scholarly research and sponsored projects. Preferred specializations are Big Data/high performance computing/visualization, computer and information security, software engineering. Applicants must hold a Ph.D. in CS (or closely related field) or complete it by the start date in August 2013, and be committed to excellence in teaching, scholarly research, and external funding. BGSU is located about an hour from Detroit airport, and the area offers excellent quality of life. BGSU is an AA/EEO. Email cssearch2013@cs.bgsu.edu a letter of interest, along with the name, title, email, and postal addresses for three professional references, curriculum vitae, official or unofficial transcripts by January 13, 2013. For finalists, three current letters of reference, official transcript of the highest degree, and background check are required. For details, go to <http://www.bgsu.edu/departments/compsci/jobs>

Bradley University Assistant Professor Department of Computer Science and Information Systems

The CS&IS Department invites applications for a tenure track Assistant Professor position starting in August 2013. **The positions require that a PhD in Computer Science or a closely related field is completed prior to the start date.** We will consider strong candidates from all areas of computer sciences but applicants with expertise in mobile computing, architecture, security, and gaming are especially encouraged.

The successful candidate is expected to develop into an outstanding teacher at the undergraduate and graduate levels, establish and maintain an active and rigorous research program, actively seek funding in support of his/her research activities, and involve students in his/her research activities. The teaching load is three courses per semester. Some teaching experience is preferred.

Initial screening will begin on December 10, 2012 and will continue until the position is filled.

The CS&IS Department at Bradley University currently consists of 9 full time tenure/tenure-track faculty professorial positions and one lecturer position. We currently offer bachelors and masters degrees in both Computer Science and Computer Information Systems.

To apply, send a letter of application, a curriculum vita, transcripts, and at least three letters of reference to Chair Search Committee, Department of CS&IS, Peoria, IL 61625 (e-mail: jawalter@bradley.edu).

Employment with Bradley University is contingent upon satisfactory completion of a criminal background check. Bradley University is an EEO/AA Employer.

California Institute of Technology Tenure-track Faculty

The Caltech Computing and Mathematical Sciences (CMS) Department (<http://www.cms.caltech.edu>) invites applications for a tenure-track faculty position. The CMS department is a unique environment where innovative, multidisciplinary, and foundational research is conducted in a collegial atmosphere. We are looking for candidates in Applied Mathematics, Computer Science, and Control & Dynamical Systems, who have demonstrated exceptional promise through innovative research with strong potential connections to other mathematical, natural, and engineering sciences. A commitment to high-quality teaching and mentoring is also expected.

The initial appointment at the assistant-professor level is for four years and is contingent upon the completion of a Ph.D. degree in a CMS-related field. To ensure the fullest consideration, applicants are encouraged to have all their application materials on file by January 1, 2013.

Please view the application instructions and apply on-line at http://www.eas.caltech.edu/positions/cms_broad/

Questions about the application process may be directed to: info@cms.caltech.edu.

Caltech is an Equal Opportunity/Affirmative

Action Employer. Women, minorities, veterans, and disabled persons are encouraged to apply.

Colorado School of Mines Faculty Openings

The Department of Electrical Engineering and Computer Science invites applications for multiple faculty positions.

G.A. Dobelman Chair in Electrical Engineering
Assistant Professor Positions in EECS
Teaching Associate Professor

For complete job announcements and directions on how to apply for these openings, visit: <http://inside.mines.edu/HR-Academic-Faculty>

Colorado State University Department of Computer Science Tenure Track Assistant or Associate Professor

Colorado State University is accepting applications for a tenure-track assistant or associate professor in Computer Science, beginning fall 2013. Only candidates in bioinformatics/ computational biology will be considered. This position is part of a university-wide effort to recruit additional faculty in biosciences and bioinformatics. More information may be viewed at <http://www.cs.colostate.edu>.

Applications must be received by **January 11, 2013**. Submit materials at <http://cns.natsci.co>

lostate.edu/employment/CompSci/. Application materials of semifinalist candidates, including letters of reference, will be made available for review by the entire faculty of the Department of Computer Science.

CSU is an EO/EA/AA employer. Colorado State University conducts background checks on all final candidates.

Connecticut College Tenure-Track CS Professor

Connecticut College is seeking candidates with research interests in visual aspects of computing, including graphics, visualization, human/computer interaction, virtual/augmented reality, animation, and other fields related to visual media. See details at cs.conncoll.edu/tenure-track-position.htm for this tenure-track position to begin in August.

Dartmouth College Roth Family Distinguished Professorship

The Department of Computer Science at Dartmouth College invites applications for the inaugural Roth Family Distinguished Professorship. We seek candidates with a strong academic or industry track record in the general area of Digital Arts (including, but not limited to, Visualization; Human-Computer Interaction; Tangible Computing; Computer Graphics; Computer Vision, Music, or Audition; and Media Arts and Sciences).



Yale School of Engineering & Applied Science Department of Electrical Engineering

Junior Search in Communications and Networking at Yale University

Yale University's Electrical Engineering Department invites applications from qualified individuals for a tenure-track, non-tenured faculty position in the area of communications and networking. Subfields of interest include wireless communications, networking, signal processing, network optimization, network economics, machine learning, and network science. All candidates should be strongly committed to both teaching and research and should be open to collaborative research. Candidates should have distinguished records of research accomplishments and should be willing and able to participate in shaping Yale's expanding program in electrical engineering. Yale University is an Affirmative Action/Equal Opportunity Employer. Yale values diversity among its students, staff, and faculty and strongly welcomes applications from women and under represented minorities. The review process will begin November 15, 2012. Applicants should include a CV, a research statement, a teaching statement and submit to <http://academicjobsonline.org/>.

Senior Search in Communications and Networking at Yale University

Yale University's Electrical Engineering Department invites applications from qualified individuals for a tenured faculty position in the area of communications and networking. Subfields of interest include wireless communications, networking, signal processing, network optimization, network economics, machine learning, and network science. All candidates should be strongly committed to both teaching and research and should be open to collaborative research. Candidates should have distinguished records of research accomplishments and should be willing and able to take the lead in shaping Yale's expanding program in electrical engineering. Yale University is an Affirmative Action/Equal Opportunity Employer. Yale values diversity among its students, staff, and faculty and strongly welcomes applications from women and under represented minorities. The review process will begin November 15, 2012. Applicants should include a CV, a research statement, a teaching statement and submit to <http://academicjobsonline.org/>.

Senior Position in Computer Engineering at Yale University

Yale University's Electrical Engineering Department invites applications from qualified individuals for a tenured faculty position in computer engineering. Subfields of interest include systems on a chip, embedded systems, VLSI, design automation, energy-efficient computing, low-power circuits, verification, networked systems, mobile computing, sensor networks, and biodevices. All candidates should be strongly committed to both teaching and research and should be open to collaborative research. Candidates should have distinguished records of research accomplishments and should be willing and able to take the lead in shaping Yale's expanding program in computer engineering. Yale University is an Affirmative Action/Equal Opportunity Employer. Yale values diversity among its students, staff, and faculty and strongly welcomes applications from women and under represented minorities. The review process will begin on November 15, 2012. Applicants should include a CV, a research statement, a teaching statement and submit to <http://academicjobsonline.org/>.

Candidates at the level of full or associate professor will be considered.

Dartmouth is home to a growing program in the Digital Humanities and Arts with affiliated faculty and students in Computer Science, English, Film & Media Studies, Mathematics, Music, Psychology, Studio Art, and Theater. In the coming years we expect continued investment in the Digital Arts. The Roth Family Distinguished Professor will help shape what is expected to be a leading undergraduate and graduate program in the Digital Arts.

The Computer Science department (www.cs.dartmouth.edu) is home to 15 tenured and tenure-track faculty members whose research encompasses the areas of digital arts, graphics, vision, algorithms, theory, systems, security, robotics, and computational biology. The Computer Science department is in the School of Arts and Sciences, and it has strong Ph.D. and M.S. programs and outstanding undergraduate majors. The department hosts the undergraduate minor in Digital Arts, and has strong collaborations with Dartmouth's programs in Digital Humanities, Neuroscience, Digital Music, Computational Genetics, the Tuck School of Business, the Geisel School of Medicine, and is home to the interdisciplinary Neukom Institute for Computational Science.

Dartmouth College, a member of the Ivy League, is located in Hanover, New Hampshire (on the Vermont border). Dartmouth has a beautiful, historic campus, located in a scenic area on the Connecticut River. With an even distribution of male and female students and over a quarter of

the undergraduate student population members of minority groups, Dartmouth is committed to diversity and encourages applications from women and minorities. Dartmouth College is an equal opportunity and affirmative action employer.

Applicants are invited to send their CV, research statement, teaching statement, and names of at least four references. Please arrange to have at least one letter writer comment about teaching. All material and inquiries should be sent to roth2013@cs.dartmouth.edu. Application review will start on January 14, 2013.

Applicants should arrange to have letters of recommendation sent directly by the recommender, either by email or physical mail, to:

Roth Family Professorship Search
Department of Computer Science
Dartmouth College
6211 Sudikoff Laboratory
Hanover, NH 03755
roth2013@cs.dartmouth.edu

Dartmouth College is an equal opportunity/affirmative action employer and encourages applications from women and members of minority groups.

Endicott College **Assistant Professor of Computer Science**

Seeking a dynamic full time faculty member with a passion for undergraduate education in Computer Science to help advance and grow the Computer Science program with the development of

new initiatives and curricula; will teach a variety of courses in Computer Science, advise students, participate in school and college-wide service, and engage in undergraduate focused research.

Doctorate in Computer Science or a related field; outstanding teaching record in higher education; research program experience involving undergraduate students; experience mentoring students; ability to work as part of an interdisciplinary team; vision and drive to grow our department, attract new students, initiate and maintain industrial liaisons, and expand Endicott's presence in the community.

APPLY FOR THIS JOB

Email Address:

humanresources@endicott.edu

Fax: 978-927-0084

The George Washington University **Department of Computer Science** **One Faculty Position**

The Department of Computer Science at The George Washington University is seeking applicants for one tenure-track/tenured faculty position at the rank of Assistant/Associate Professor, starting Fall 2013. Applicants in all areas of Computer Science are welcome. Applicants who will enhance or complement our current strengths are especially encouraged to apply.

All applicants must have a Ph.D. degree in Computer Science or a closely related field. The successful candidate must complete all degree requirements for Ph.D. in Computer Science



Florida International University is a multi-campus public research university located in Miami, a vibrant, international city. FIU is recognized as a Carnegie engaged university. Its colleges and schools offer more than 180 bachelor's, master's and doctoral programs in fields such as computer science, engineering, international relations, architecture, law, and medicine. As one of South Florida's anchor institutions, FIU is worlds ahead in its local and global engagement, finding solutions to the most challenging problems of our time. FIU emphasizes research as a major component of its mission and enrolls 48,000 students in two campus and three centers including FIU Downtown on Brickell and the Miami Beach Urban Studios. More than 160,000 alumni live and work in South Florida. For more information about FIU, visit <http://www.fiu.edu/>.

The School of Computing and Information Sciences seeks exceptionally qualified candidates for multiple tenure-track and tenured faculty positions at all levels as well as non-tenure track faculty positions at the level of Instructor.

Tenure track/tenured positions (Job ID # 505004)

We seek well-qualified candidates in all areas of Computer Science and researchers in the areas of programming languages, compilers, databases, information retrieval, computer architecture, scientific computing, big data, natural language processing, computational linguistics, health informatics, and robotics, are particularly encouraged to apply. Preference will be given to candidates who will enhance or complement our existing research strengths.

Ideal candidates for junior positions should have a record of exceptional research in their early careers. Candidates for senior positions must have an active and proven record of excellence in funded research, publications, and professional service, as well as a demonstrated ability to develop and lead collaborative research projects. In addition to developing or expanding a high-quality research program, all successful applicants must be committed to excellence in teaching at both graduate and undergraduate levels. An earned Ph.D. in Computer Science or related disciplines is required.

Non-tenure track instructor positions (Job ID# 505000)

We seek well-qualified candidates in all areas of Computer Science and Information Technology. Ideal candidates must be committed to excellence in teaching a variety of courses at the undergraduate level. A graduate degree in Computer Science or related disciplines is required; significant prior teaching and industry experience and/or a Ph.D. in Computer Science is preferred.

Florida International University (FIU), the state university of Florida in Miami, is ranked by the Carnegie Foundation as a comprehensive doctoral research university with high research activity. The School of Computing and Information Sciences (SCIS) is a rapidly growing program of excellence at the University, with 36 faculty members and over 1,500 students, including 75 Ph.D. students. SCIS offers B.S., M.S., and Ph.D. degrees in Computer Science, an M.S. degree in Telecommunications and Networking, and B.S., B.A., and M.S. degrees in Information Technology. SCIS has received approximately \$17.5M in the last four years in external research funding, has six research centers/clusters with first-class computing infrastructure and support, and enjoys broad and dynamic industry and international partnerships.

HOW TO APPLY:

Applications, including a letter of interest, contact information, curriculum vitae, academic transcript, and the names of at least three references, should be submitted directly to the FIU Careers website at <https://jobsearch.fiu.edu> refer to **Job ID# 505004** for tenure-track or tenured positions and to **Job ID# 505000** for instructor positions. The application review process will begin on January 7, 2013, and will continue until the position is filled. Further information can be obtained from the School website <http://www.cis.fiu.edu> or by e-mail to recruit@cis.fiu.edu.

*FIU is a member of the State University System of Florida
and is an Equal Opportunity,
Equal Access Affirmative Action Employer.*

or a closely related field by August 15, 2013. Applicants must have outstanding research credentials, demonstrated ability or potential for attracting research funding, and demonstrated teaching excellence or potential.

Application Procedure: To be considered, applicants must submit an application online at <http://www.gwu.jobs/postings/11888>. The application should contain a brief cover letter indicating the area and rank of interest, a curriculum vitae, research and teaching statements, names and contact information for at least three references. When requested, reference letters should be emailed to cssearch@gwu.edu. There is no deadline, but review of applications will begin December 2, 2012 and continue until the position is filled. Only complete applications will be considered.

The Department of Computer Science offers B.S., B.A., M.S. and Ph.D. degree programs in Computer Science, and an M.S. degree program in Cybersecurity. Embarked on a rapid growth process, the Department has hired nine tenure-track professors in the past four years, plans to continue hiring for the next several years, and will soon be housed in a \$275M, 500,000 sq. ft., state-of-the-art building along with other engineering and natural science departments. Currently, the Department has 20 regular faculty members, numerous affiliated and adjunct faculty members, and over 650 students. Its educational and research programs are in security, systems, networks, graphics with biomedical applications, AI, robotics, machine learning, NLP, imaging, and search, with funding from various agencies such as NSF, DOD, NIH and others. Part of a mid-size

private university, the Department prides itself on excellent research, quality education, and low student-teacher ratio. Its location in the heart of the nation's capital provides the faculty close access to many Federal funding agencies and research labs, and affords the whole university community unique cultural and intellectual opportunities. For further information, please refer to <http://www.cs.gwu.edu>.

The George Washington University is an equal opportunity/affirmative action employer.

The George Washington University Assistant/Associate Professor in Electrical and Computer Engineering

The Department of Electrical and Computer Engineering at The George Washington University invites applications for a tenure-track position at the Assistant/Associate Professor Rank in the area of High-Performance Computing or Computer Hardware Systems for Biomedical Engineering, Bioinformatics, or Computational Biology. Successful candidates may start as early as the Fall semester 2013.

The department has well-established graduate and undergraduate programs in Electrical, Biomedical, and Computer Engineering. In addition, the University has initiatives designating Biomedical Engineering, Energy, High Performance Computing, and Nano-Technology as areas of excellence. Under construction is a 500,000 sq. ft. Science and Engineering Hall that will open in 2015 and house state-of-the-art clean rooms,

imaging facilities, and research and instructional laboratories (<http://www.gwu.edu/science-and-engineeringhall/>). The George Washington University is located in the nation's capital near many federal funding agencies and government research laboratories.

Responsibilities: The successful candidate for this position will be expected to develop externally-sponsored research programs, supervise graduate students, and contribute to the teaching and academic advising of students at all levels. Teaching responsibilities include developing and delivering both undergraduate and graduate courses and laboratories in areas that may include some combinations of digital design, embedded systems, biomedical engineering, and computational biology and bioinformatics. The successful candidate will undertake interdisciplinary research in computer engineering and biomedical engineering aimed at using high-performance computing, reconfigurable computing, and embedded systems to advance biomedical engineering in areas such as computer and robotics assisted surgery, computational biology, imaging, and/or bioinformatics.

Basic Qualifications: Applicants for the Associate Professor Rank are expected to have a Ph.D. in Biomedical, Computer, or Electrical Engineering, or a closely related field. They should have strong records of attracting research funding, peer-reviewed publications, and teaching, as evidenced by teaching evaluations and recognition of teaching excellence.

Applicants for the Assistant Professor Rank must complete all degree requirements for the

Faculty positions in School of Information Science and Technology, ShanghaiTech University

The School of Information Science and Technology (SIST) at the inaugural ShanghaiTech University offers both undergraduate and graduate degree programs in all primary areas of information science and technology. ShanghaiTech is founded as a world-class research university aimed at training future generations of scientists, entrepreneurs, and technological leaders in China. ShanghaiTech is located in the Zhangjiang High-Tech Park in the cosmopolitan city of Shanghai. We are committed to building an open and free academic environment by seeking excellence through diversity and inclusion.

SIST is recruiting highly qualified candidates to fill multiple tenure-track as well as tenured positions at all levels to join us in this exciting inaugural phase. Candidates should have an exceptional track record or demonstrated strong potential in active research areas of information science and technology or closely related fields. Besides maintaining an active research program, successful candidates are also expected to contribute substantially to the educational missions of undergraduate and graduate programs within SIST.

Academic Disciplines: We seek faculty candidates in all cutting edge areas of Information science and technology, with special focus on: advanced futuristic computer architecture and technology, nano-scale electronics, ultra-high speed and low power circuits, intelligent multimedia and integrated signal processing systems, next-generation computer systems, computational foundations, big data, data mining, visualization, computer vision, bio-computing, smart energy devices and systems, highly-scalable and multi-service heterogeneous networking, as well as various inter-disciplinary areas involving the foundation and applications of information science and technology.



- Qualifications:** Candidates must demonstrate
- a strong interest in undergraduate and graduate education;
 - well developed research plans and demonstrated strength;
 - Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, or closely related field);
 - A minimum relevant research experience of 4 years.

Applications: Qualified applicants are invited to submit a cover letter, a 2-page research plan, a CV including copies of up to 3 most significant publications, and the names of three referees to: sist@shanghaitech.edu.cn.

Mailing Address: School of Information Science and Technology, ShanghaiTech University Building 2, 319 Yueyang Road, Shanghai 200031, China

ShanghaiTech University Website: <http://www.shanghaitech.edu.cn>

Deadline: December 31st, 2012 (Late submissions may be considered until all positions are filled.)

Compensation and Benefits: Salary and startup fund are competitive, commensurate with experience and academic accomplishment. ShanghaiTech also offers a comprehensive benefit package to employees and their eligible dependents, including housing benefits.

Ph.D. in Biomedical, Electrical, or Computer Engineering, or a closely related field, before September 1, 2013, to be appointed as an assistant professor. In addition, applicants must: 1) demonstrate strong research potential, as evidenced by peer-reviewed publications; and 2) demonstrate substantial potential to attract extramural research funding.

Applicants are expected to have significant experience in designing and implementing experimental cross-disciplinary systems, and to have a demonstrated commitment to the responsibilities listed above.

How to Apply: To be considered, applicants must complete an online faculty application specific to the rank of interest and upload (i) a brief statement of interest, (ii) a curriculum vitae, (iii) a statement of research and teaching interests, (iv) a sample of three publications, (v) a representative sample of course evaluations or teaching assessments, if applicable, and (vi) complete contact information for at least five references.

For Associate Rank, please visit
<https://www.gwu.jobs/postings/11953>.

For Assistant Rank, please visit
<https://www.gwu.jobs/postings/11952>.

Review of applications will begin on 1/1/2013 and will continue until the position is filled.

The George Washington University is an Equal Opportunity/Affirmative Action Employer.

Georgia Institute of Technology Tenure-Track Faculty Positions at the School of Computer Science, Georgia Tech School of Computer Science

The School of Computer Science in the College of Computing at the Georgia Institute of Technology invites applications for tenure-track faculty positions. We are primarily seeking junior-level candidates at the **Assistant Professor** level. Truly exceptional candidates at the Associate and Full Professor level may also be considered. Applicants should have completed or be near completion of a Ph.D. in computer science or a related field and should demonstrate potential for excellence in research and teaching. The School of Computer Science, one of three schools in the College of Computing, focuses on research that makes computing and communication smart, fast, reliable, and secure, with research groups in computer architecture, databases, machine learning, networks, programming languages, security, software engineering, systems, and theory. We seek candidates whose research addresses the future challenges of computing either by adding to or enhancing the current research areas of the school. Candidates who bridge research areas within the school or between the school and other disciplines are particularly welcome.

Applications will be considered until open positions are filled. However, to receive full consideration, applications should be submitted by December 10, 2012. Application materials should include a full academic CV, teaching and research statements, a list of at least three references, and

up to three publications. Applicants are encouraged to clearly identify in their cover letter the area(s) that best describe their research interests. All applications must be submitted online. More information about the School of Computer Science and application instructions are available at <http://scs.gatech.edu>.

Georgia Tech is an Affirmative Action/Equal Opportunity Employer.

Harvard University School of Engineering and Applied Sciences Assistant Professor

The Harvard School of Engineering and Applied Sciences (SEAS) seeks applicants for a position at the level of tenure-track assistant professor in Computer Science, with an expected start date of July 1, 2013.

This is a broad search, and we welcome outstanding applicants in all areas of computer science. This includes applicants whose research and interests connect to such areas as computational science, engineering, health and medicine, or the social sciences.

The Computer Science program at Harvard University benefits from outstanding undergraduate and graduate students, an excellent location, significant industrial collaboration, and substantial support from the School of Engineering and Applied Sciences. Information about Harvard's current faculty, research, and educational programs is available at <http://www.seas.harvard.edu/>.

LSU

DIRECTOR OF THE SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

The College of Engineering's newly formed School of Electrical Engineering and Computer Science (EECS) at Louisiana State University, Baton Rouge (LSU) seeks applicants and nominations for the position of school director. An individual meeting college-wide criteria will also be appointed to the endowed M.B. Voorhies Professorship.

About the School: The School of Electrical Engineering and Computer Science was formed from two separate departments in 2012. It resides in the College of Engineering and includes approximately 40 faculty members distributed across two divisions: the Division of Electrical and Computer Engineering (ECE) and the Division of Computer Science and Engineering (CSE). ECE offers BS degrees in electrical engineering and computer engineering programs, MS and Ph.D. degrees in electrical engineering. CSE offers a BS degree in computer science, an MS degree in system science, and a Ph.D. degree in computer science. All undergraduate programs in the school are accredited by ABET. The College of Engineering at LSU is home to eleven degree programs, has the second largest student enrollment on campus and ranks 7th in the nation for the number of undergraduate degrees granted. LSU is the flagship university in Louisiana, is classified by Carnegie Research Foundation as RU/VH (research university, very high research activity), and has a current enrollment exceeding 28,000 students. Baton Rouge, located on the east bank of the Mississippi River, has a metropolitan population of approximately 800,000 and is the center of major industrial, petrochemical, medical, government, and research activity.

Responsibilities: The director of the School of EECS will be responsible for advancing research, teaching, and outreach activities, as well as managing school needs, resources, budgets, and personnel. This individual will be expected to advance the school in research activity, fundraising, and improving visibility nationally and internationally. He or she is expected to exhibit a commitment to diversity, have an appreciation of the diverse research and teaching activities in the school, and take a leadership role in interactions with alumni and industry. The director reports directly to the dean of the College of Engineering and is the primary liaison between the school and the college. Associate chairs of ECE and CSE will coordinate the division level administrative and academic functions and report to the school director.

Qualifications: The newly formed School of EECS is looking for a director with the vision to leverage the existing potential of this school, as well as develop new opportunities. The position requires an earned doctorate from an appropriate accredited institution in electrical engineering, computer engineering, computer science, or a closely related discipline. We invite applications from candidates with outstanding records in scholarly research and demonstrated leadership skills, including the ability to communicate effectively with students, faculty, alumni, industry partners, and the community.

Application: Candidates should include a complete CV and a list of at least three professional references with the electronic application materials. An offer of employment is contingent on a satisfactory pre-employment background check. Applications will be reviewed beginning January 1, 2013 and will continue until a suitable candidate is selected. Apply online at www.lsu.systemcareers.lsu.edu, **Position #026403**. LSU welcomes nominations of candidates. Letters of nomination may be e-mailed to the search committee chair at karsten@lsu.edu.

LSU IS AN EQUAL OPPORTUNITY/EQUAL ACCESS EMPLOYER

Candidates are required to have a PhD or an equivalent degree by the expected start date. In addition, we seek candidates who have an outstanding research record and a strong commitment to undergraduate teaching and graduate training.

Required application documents include a cover letter, CV, a statement of research interests, a teaching statement, up to three representative papers, and names and contact information for at least three references. Applicants will apply on-line at <http://academicpositions.harvard.edu/postings/4324>.

We encourage candidates to apply by December 1, 2012, but will continue to review applications until the positions are filled. Harvard is an Equal Opportunity/Affirmative Action Employer. Applications from women and minority candidates are strongly encouraged.

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 2012, we were ranked 13th among all Computer Science

and Engineering Departments worldwide according to QS World University Ranking, and 26th according to Academic Ranking of World Universities.

The Department will have at least one tenure-track faculty opening at Assistant Professor/Associate Professor/Professor levels for the 2013-2014 academic year. We are looking for candidates with interests in **Security**, or **Systems**. Strong candidates in core computer science and engineering research areas will also be considered. Applicants at Assistant Professor level should have an earned PhD degree and demonstrated potential in teaching and research.

Salary is highly competitive and will be commensurate with qualifications and experience. Fringe benefits include medical and dental benefits and annual leave. Housing benefits 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 **28 February 2013**. 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.)

Marist College **Assistant Professor of CS/ITS**

Marist College, located in the historic Hudson River Valley of New York, invites applications for tenure-track positions in Information Technology & Systems and Computer Science. We welcome candidates who have teaching and research interests in the following areas: data and information security, information decision systems, software development, data management, cloud computing, business analytics, and systems analysis and design.

Candidates must have a doctoral degree in Computer Science, Information Technology, Information Systems, or a closely related field. ABDs in an appropriate field will be considered. Evidence of excellence in teaching and scholarly work is required; consulting experience is highly desirable.

Apply URL: https://jobs.marist.edu/view_position.php?postingid=232

New York University/Courant Institute of Mathematical Sciences **Department of Computer Science** **Clinical Assistant Professor**

The Computer Science Department at New York University has an opening for a Clinical Assistant Professor position to start September 1, 2013, subject to budgetary and administrative approval. This is a full-time non-tenured, non-tenure-track three-year contract faculty position



Edgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Assistant Professorships (Tenure Track) in Computer Science

The Department of Computer Science (www.inf.ethz.ch) at ETH Zurich invites applications for assistant professorships (Tenure Track) in the areas of:

- Computer Systems
- Software Engineering
- Information Systems (with emphasis on Big Data)

For candidates with exceptional research accomplishments also applications for a full professorship will be considered.

The department offers a stimulating and well-supported research and teaching environment. Collaboration in research and teaching is expected both within the department and with other groups of ETH Zurich and related institutions.

Applicants should have internationally recognized expertise in their field and pursue research at the forefront of Computer Science. Successful candidates should establish and lead a strong research program. They will be expected to supervise Ph.D. students and teach both undergraduate level courses (in German or English) and graduate level courses (in English).

Assistant professorships have been established to promote the careers of younger scientists. The initial appointment is for four years with the possibility of renewal for an additional two-year period and promotion to a permanent position.

Your application should include your curriculum vitae, a list of publications, a statement of research and teaching interests and the names of at least three referees. The letter of application should be addressed to the **President of ETH Zurich, Prof. Dr. Ralph Eichler**. **The closing date for applications is 15 January 2013**. ETH Zurich is an equal opportunity and affirmative action employer. In order to increase the number of women in leading academic positions, we specifically encourage women to apply. ETH Zurich is further responsive to the needs of dual career couples and qualifies as a family friendly employer. **Please apply online at www.facultyaffairs.ethz.ch.**

which is potentially renewable. The main duty is to teach three courses during each of the fall and spring semesters in the department's undergraduate program and additionally to participate in curricular development, program administration, and other educational activities. Applicants should have an M.S. in Computer Science or a related field, or an M.P.S. in Interactive Telecommunications. To apply, please arrange for a CV and for three letters of recommendation to be sent by email to jobs@cs.nyu.edu. To guarantee full consideration, complete applications should be received by January 15, 2013. However, all candidates will be considered to the extent feasible until the position is filled. NYU is an Equal Opportunity/Affirmative Action Employer.

North Dakota State University **Assistant/Associate Professor**

Department of Computer Science at North Dakota State University seeks to fill a tenure-track Assistant/Associate Professor position starting Fall of 2013. PhD required. NDSU offers degrees in Computer Science and Software Engineering. Research and teaching excellence is expected. Normal teaching loads are 3 courses per year. Salary is competitive.

The Department has 18 Faculty in diverse areas, approximately 200 graduate (PhD and MS) students and 300 BS/BA students. NDSU is a Carnegie research extensive class institution.

Fargo is a clean, growing metropolitan area of 250,000 that consistently ranks very high in na-

tional quality-of-life surveys. We have low levels of crime and pollution, excellent schools, short commutes and proximity to the Minnesota lakes country. The community has a symphony, an opera, a domed stadium, a community theater, three universities and many other opportunities for advancement, recreation and entertainment.

Minimum Qualifications:

Ph.D in Computer Science or a closely related area. Experience or other evidence of potential for excellence in research, undergraduate and graduate teaching and service.

Effective oral and written communication skills.

Qualifications preferred but not required:

Experience or other evidence of potential for excellence in research in software engineering, networks, security, or a related area. The capability for teaching a variety of courses in computer science.

To Apply: see <https://jobs.ndsu.edu/>

For additional information see
<http://cs.ndsu.edu/positions.htm>

Review of applications begins January 1, 2013. Applications will be accepted until position is filled.

North Dakota State University is an Equal Opportunity/AA employer.

This position is exempt from North Dakota Veterans' Preference requirements.

Oregon State University **School of Electrical Engineering and** **Computer Science** **Assistant/Associate/Full Professor tenure-track Position in Computer Science**

The School of Electrical Engineering and Computer Science at Oregon State University invites applications for several tenure-track positions in Computer Science. We seek strong candidates with a commitment to quality teaching and with research strengths in the areas of programming languages, databases, software engineering, computer systems, computer security and privacy, and algorithms. Applicants should demonstrate a strong commitment to collaboration with other research groups in the School of EECS, with other departments at Oregon State, and with other universities.

The School of EECS supports a culture of energetic collaboration, and the faculty are committed to excellence in both education and research. With 42 tenure/tenure-track faculty, we enroll 180 Ph.D., 120 MS and 1400 undergraduate students. OSU is recognized for its "very high research activity" by the Carnegie Foundation for the Advancement of Teaching. The School of EECS is housed in the Kelley Engineering Center, a green building designed to support collaboration among faculty and students across campus. Corvallis is a small college town renowned for its high quality of life.

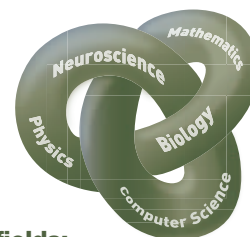
https://jobs.oregonstate.edu/applicants/jsp/shared/search/SearchResults_css.jsp

**CAMPUS
VISIT DAY**
November 24,
2012

CALL FOR PhD STUDENTS

The Graduate School at IST Austria invites applicants from all countries to its PhD program. IST Austria is a new institution located on the outskirts of Vienna dedicated to cutting-edge basic research in the natural sciences and related disciplines. The language at the Institute and the Graduate School is English.

The PhD program combines advanced coursework and research, with a focus on Biology, Computer Science, Mathematics, Physics, Neuroscience, and interdisciplinary areas. IST Austria offers internationally competitive PhD salaries supporting 4–5 years of study. Applicants must hold either a BS or MS degree or equivalent.



The Institute offers PhD students positions in the following fields:
Biology | Computer Science | Mathematics | Physics | Neuroscience

For a list of faculty members please visit www.ist.ac.at. For further information and access to the online application please consult www.ist.ac.at/gradschool. For inquiries, please contact gradschool@ist.ac.at. For students wishing to enter the program in the fall of 2013, the deadline for applications is January 15, 2013.

IST Austria is committed to Equality and Diversity.

IST AUSTRIA
Institute of Science and Technology



Faculty Positions Available in the Department of Computer Science

The Department of Computer Science (CS@VT) at Virginia Tech seeks applicants for tenure-track faculty positions in the following areas:

Associate Professor in Cybersecurity

The successful candidate will contribute to the research and graduate programs in the National Capital Region (NCR) and collaborate with faculty at Virginia Tech's campus in Blacksburg, VA. This position adds to the collaboration in cybersecurity between the Department of Computer Science and the Bradley Department of Electrical and Computer Engineering (ECE). Candidates should have research interests in systems and network security, trustworthy systems, software security, information security and privacy, or other topics relevant to national critical infrastructure. Candidates should have a record appropriate to an associate professor rank in scholarship, leadership, and interdisciplinary collaboration in cybersecurity. Ideal candidates combine cybersecurity with existing departmental strengths. See www.cs.vt.edu/FacultySearch for additional information.

The department collaborates with ECE in the newly established Ted and Karyn Hume Center for National Security and Technology (www.cnst.ictas.vt.edu) and the NCR Program in Information Assurance with an associated Executive Master's degree and a graduate certificate program. The NCR campus (www.ncr.vt.edu) is located near the Washington D.C./Falls Church area and houses the Virginia Tech Research Center (www.ncr.vt.edu/arlington) in Arlington, VA, enabling significant research opportunities in cyber-security. The new facility is within walking distance of the NSF, ONR, AFOSR, DARPA, and other agencies.

Applications must be submitted online to <https://jobs.vt.edu> reference posting #0122357 or use this direct link (listings.jobs.vt.edu/applicants/Central?quickFind=196077). Applicant screening will begin December 31, 2012 and continue until the position is filled. Inquiries should be directed to **Dr. Dennis Kafura, Search Committee Chair**, kafura@cs.vt.edu.

Assistant Professor in Artificial Intelligence/Machine Learning Blacksburg, VA

Full-time tenure-track position, at the rank of Assistant Professor, from candidates with expertise in artificial intelligence having specific emphasis on machine learning or reasoning under uncertainty. The department is in the process of making multiple hires over multiple years in this area. Candidates should have a record of scholarship, leadership, and collaboration in computing and interdisciplinary areas; demonstrated ability to contribute to teaching at the undergraduate and graduate levels in AI and related subjects; sensitivity to issues of diversity in the campus community; and the skills to establish and grow a multidisciplinary research group. Early applications are encouraged. Applications must be submitted online to <https://jobs.vt.edu> for posting #0122414.

The department is home to the Discovery Analytics Center (dac.cs.vt.edu), a university-wide effort that brings together faculty with strengths in machine learning, big data, and data mining applied to problems of national interest. There also are rich opportunities in a highly collaborative department with strengths in HCI, HPC, computational biology and bioinformatics, information retrieval, software engineering, CyberArts, and CS education. Beyond the department, there are opportunities for collaboration in machine learning with faculty in ECE and Statistics. Research on security and personal health informatics is possible in collaboration with the VT-Carilion Research Institute associated with the VT-Carilion School of Medicine. Applicant screening will begin December 31, 2012 and continue until the position is filled. Inquiries should be directed to **Dr. Doug Bowman, AI/ML Search Committee Chair**, [bowman@vt.edu](mailto:b Bowman@vt.edu).

Assistant, Associate, or Full Professor in Systems Blacksburg, VA

Candidates with research breadth and depth across several areas of computer systems, including architecture, operating systems, and networking are sought and researchers in the areas of compilers, run-time systems, and parallel and distributed systems are especially encouraged to apply. Candidates should have a record of scholarship and collaboration in computing and interdisciplinary areas; demonstrated ability to contribute to teaching at the undergraduate and graduate levels; and sensitivity to issues of diversity in the campus community. Applications must be submitted online to <https://jobs.vt.edu> for posting #0122413. Applicant screening will begin December 31, 2012 and continue until the position is filled. Inquiries should be directed to **Dr. Kirk W. Cameron, Search Committee Chair**, cameron@cs.vt.edu.

The Department of Computer Science (CS@VT) has 35 tenure-track research oriented faculty including 12 NSF CAREER award winners, Ph.D. production among the top 30 in the USA, and annual research expenditures exceeding \$6.5 million. CS@VT is in the College of Engineering whose undergraduate program was ranked 16th among all accredited engineering schools, and sixth among public universities and whose graduate program was ranked 24th by *US News & World Report* in fall 2012. In 2010, CS@VT was ranked 5th in the recruiting quality of computer science undergraduate majors by the *Wall Street Journal*. Recently, the department has attracted high-profile research funding including several multi-million dollar awards from diverse sources, e.g., IARPA, NSF, DOE, and ARO.

Salary for suitably qualified applicants is competitive and commensurate with experience.

Virginia Tech is an equal opportunity/affirmative action institution.

Peking University School of EECS Tenure-Track Positions

The School of EECS at Peking University invites applications for tenure-track positions in the areas of energy efficient computing (including but not limited to energy-efficient architectures, communication, compilation, and system software) and applications (such as smart grid, mobile computing, sensor networks, and hardware acceleration of computing-intensive applications). These positions are associated with the Center for Energy-Efficient Computing and Applications (<http://ceca.pku.edu.cn>), which offers a new level of startup and compensation packages. Applications from distinguished candidates at senior levels are also encouraged. To apply, please email the resume, statements of research and teaching, and at least three names for references to ceca_recruiting@pku.edu.cn. Applications received by January 15, 2013 will be given full consideration. Early submissions are encouraged (the first of set of interviews will be in December 2012).

The Pennsylvania State University Department of Computer Science and Engineering Faculty Position Vacancies

Applications are invited for several tenure-track faculty positions at all ranks. Outstanding candidates in all areas of Computer Science and Engineering will be considered. Areas of particular interest are: computer architecture and embedded/VLSI systems; all aspects of cloud systems; and data mining with an emphasis on big data. Positions related to the last two areas are part of a Penn State Cyber-science Cluster Hire in interdisciplinary computation- and data- enabled science and engineering (see <http://www.ics.psu.edu/hire.html>). This cross-college initiative coordinates significant resources to appoint and develop exceptional candidates who can advance algorithms, software and systems with a demonstrated commitment to participating in interdisciplinary research.

Applicants should hold a Ph.D. in Computer Science, Computer Engineering, or a closely related field and should be committed to excellence in both research and teaching. Support will be provided to the successful applicants for establishing their research programs. We encourage dual career couples to apply. Applications should be received by January 31, 2013 to receive full consideration. To apply by electronic mail, send your resume (including curriculum vitae and the names and addresses of at least three references) as a pdf file to recruiting@cse.psu.edu.

The department has 31 tenure-track faculty in the major areas of computer science and engineering. Fourteen members of our faculty are NSF Career Award recipients. Three faculty members have received the prestigious NSF PECASE Award. In recent years, our faculty received a \$35M Network Science Center Award, over \$4.5M in computing and research infrastructure and instrumentation grants from NSF, and are part of an NSF ERC and NSF IGERT team. The faculty are highly successful in obtaining funding from NSF, DARPA, ARL, DOE, DTRA, AFOSR and AOR. There are state-of-the-art research labs for computer systems, computer vision and robotics, microsystems design and VLSI, networking and security,

high performance computing, bioinformatics and virtual environments. The department offers a graduate program with 47 Masters students and 168 Ph.D. students, and undergraduate programs in computer science and computer engineering. The university is committed to growing the faculty ranks and promoting interdisciplinary research toward cyber-enabled discovery and design.

Penn State is a major research university and is ranked 3rd in the nation in industry-sponsored research. Computer science is ranked 8th in the nation in research expenditures. U.S. News and World Report consistently ranks Penn State's College of Engineering undergraduate and graduate programs in the top 15 of the nation. Penn State is ranked 5th world wide in Computer Science Citations according to Thomson-Reuters.

The university is located in the beautiful college town of State College in the center of Pennsylvania. State College has 40,000 inhabitants and offers a variety of cultural and outdoor recreational activities nearby. The university offers outstanding events from collegiate sporting events to fine arts productions. Many major population centers on the east coast (New York, Philadelphia, Pittsburgh, Washington D.C., Baltimore) are only a few hours drive away and convenient air services to several major hubs are operated by three major airlines out of State College.

Chair, Faculty Search Committee
The Pennsylvania State University
Department of Computer Science
and Engineering
Box ACM
University Park, PA 16802-6106 USA

For more information about the Department of CSE at PSU, see <http://www.cse.psu.edu>. Click here to fill out an Affirmative Action Applicant Data Card. Our search number is 015-109. You **MUST** include this search number in order to submit this form.

Penn State is committed to affirmative action, equal opportunity and the diversity of its workforce.

Swarthmore College **Tenure Track Assistant Professor**

Swarthmore College has a strong institutional commitment to excellence through diversity in its educational program and employment practices and actively seeks and welcomes applications from candidates with exceptional qualifications, particularly those with demonstrable commitments to a more inclusive society and world.

Applications are invited for a tenure track position at the assistant professor level beginning Fall semester 2013. Swarthmore College is a small, selective, liberal arts college located 10 miles outside of Philadelphia. The Computer Science Department offers majors and minors at the undergraduate level. Applicants must have teaching experience and should be comfortable teaching a wide range of courses at the introductory and intermediate level. Candidates should additionally have a strong commitment to involving undergraduates in their research. A Ph.D. in CS by or near the time of appointment is required. We are particularly interested in applicants that add breadth to our department, including the

areas of databases, networking, security, theory, compilers, and programming languages. Strong applicants in other areas will also be considered.

Priority will be given to applications received by December 15, but will be accepted until the position is filled. Applications should include a vita, teaching statement, research statement, and three letters of reference, at least two that speak to the candidate's teaching ability.

Apply URL: <http://goo.gl/LPYF2>

Contact Person: Richard Wicentowski

Email Address:

jobs2013@cs.swarthmore.edu

Phone: 610-328-8272

Fax: 610-328-8606

Texas Tech University **Assistant or Associate Professor**

The Department of Computer Science at Texas Tech University invites applications for a tenure-track position at the rank of assistant or associate professor starting in Fall 2013. Successful candidates must have a Ph.D. in computer science or a closely related field, be able to teach graduate and undergraduate courses, and perform research evidenced by scholarly publications. Successful candidates are also expected to contribute through professional and departmental services. Preference will be given to researchers in cyber security and software engineering, candidates who can teach core systems courses, and candidates with strong potential to obtain extramural funding. Applications from women and minorities are encouraged.



MILWAUKEE SCHOOL OF ENGINEERING

SOFTWARE ENGINEERING FACULTY

The Milwaukee School of Engineering invites applications for a full-time faculty position in our Software Engineering program beginning in the fall of 2013. Rank will depend on qualifications and experience of the candidate. Applicants must have an earned doctorate degree in Software Engineering, Computer Engineering, Computer Science or closely related field, as well as relevant experience in software engineering practice.

The successful candidate must be able to contribute in several areas of software engineering process and practice while providing leadership in one of the following: computer security, networks, software architecture and design, or software requirements.

MSOE expects and rewards a strong primary commitment to excellence in teaching at the undergraduate level. Continued professional development is also expected.

Our ABET-accredited undergraduate software engineering program had its first graduates in Spring 2002. Founded in 1903, MSOE is a private, application-oriented university with programs in engineering, business, and nursing. MSOE's 15+ acre campus is located in downtown Milwaukee, in close proximity to the Theatre District and Lake Michigan. Please visit our website at www.msoe.edu.

Submit all application material via email in pdf format to:
work@msoe.edu

Applicants should include a letter of application, curriculum vitae, statement of teaching interests, and names (with email and physical addresses) of at least three references.

MSOE IS AN EQUAL OPPORTUNITY/AFFIRMATIVE ACTION EMPLOYER



Tenure Track Faculty

CIS Department
Temple University

Applications are invited for tenure-track, open rank, faculty positions in the Department of Computer and Information Sciences at Temple University.

The junior position is in the software systems area, which includes

- Software Engineering and Applications,
- Database Systems, and
- Programming Languages.

The senior position for Associate or Full Professor is open to all areas of computer science/engineering. Applicants for the senior position are expected to have an outstanding track record. Please submit applications with all requested information online at <http://academicjobsonline.org>. For further information check <http://www.cis.temple.edu> or send email to search committee chair Dr. Eugene Kwatny at gkwatny@temple.edu. Review of candidates will begin on January 2, 2013 and will continue until the positions are filled. Temple University is an equal opportunity, equal access, affirmative action employer.

The Department of Computer Science currently has 14 faculty members with 252 undergraduate and 119 graduate students. Texas Tech University, with an enrollment of 32,000 students, comprises 12 academic colleges/schools and is a part of the state-supported Texas Tech University System. The university shares its campus with the TTU Health Sciences Center.

Lubbock, a city of more than 200,000, is an economic and medical center on the Texas South Plains. The area offers a low cost of living, no state income tax, short commute times, and a rich heritage of music and culture.

Review of applications will begin in January 2013 and continue until the position is filled. A letter of application, curriculum vitae, statement of proposed research, teaching statement, and three letters of reference should be submitted electronically at <http://jobs.texasstate.edu>. Please use **Requisition number 86897**. The entities of the Texas Tech University System are Equal Opportunity Employers and employ without regard to sex, race, color, national origin, religion, age, disability, genetic information, status as a disabled or Vietnam era veteran, or other protected classes.

University at Albany – SUNY **Open Rank Faculty / Computer Science Dept.**

The Computer Science Department at the **University at Albany – SUNY** is seeking candidates for an open rank tenure-track faculty position beginning Fall 2013. We seek candidates whose research involves the design and implementation

of data analytics techniques or systems that can be used in other areas of Computer Science and/or in other disciplines.

Applicants must have a Ph.D. in Computer Science or closely related discipline. Candidates for Associate or Full Professor positions must also have a strong record of funded research. For a complete job description and application procedures, visit: <http://albany.interviewexchange.com/jobofferdetails.jsp?JOBID=35201>

Questions regarding the position may be addressed to search2013@cs.albany.edu.

The Computer Science Department at UAlbany offers B.S., M.S. and Ph.D. degrees in Computer Science. For additional information about the department, please visit <http://www.cs.albany.edu>.

University of California, Los Angeles **Computer Science Department** **Tracking #0145-1213-01**

The Computer Science Department of the Henry Samueli School of Engineering and Applied Science at the University of California, Los Angeles, invites applications for tenure-track positions in all areas of Computer Science and Computer Engineering. Applications are also encouraged from distinguished candidates at senior levels. Quality is our key criterion for applicant selection. Applicants should have a strong commitment both to research and teaching and an outstanding record of research for their level of seniority. Salary is commensurate with education and experience.

UCLA is an Equal Opportunity/Affirmative Action Employer. The department is committed to building a more diverse faculty, staff and student body as it responds to the changing population and educational needs of California and the nation. To apply, please visit <http://www.cs.ucla.edu>. Faculty candidates are urged to ensure that their applications and letters of reference are received by January 1, 2013.

The University of California **at Santa Cruz** **Department of Computer Science** **Assistant/Associate/Full Professor Position**

The Department of Computer Science at the University of California, Santa Cruz, invites applications for a tenure track (Assistant) or tenured (Associate and Full Professor) faculty position starting July 1, 2013. We seek outstanding applicants in the area of games and playable media. Preferred applicants will have research and teaching experience in games and graphics, with a research presence in the graphics community and the games and/or interactive media community, though candidates with specializations in other areas of games and interactive media will be considered. Specializations in areas particularly appropriate to games and interactive media, such as real-time animation and effects, procedural content generation, and novel interface mechanisms are preferred. The campus is especially interested in candidates who can contribute to the diversity and excellence of the academic community through their research, teaching, and service.

For complete information on the position and instructions on how to apply, please visit http://apo.ucsc.edu/academic_employment/jobs/JPF00016-13.pdf. Review of applications will begin on January 4, 2013. To ensure full consideration, applications should be completed by this date. The position will remain open until filled, but not later than June 30, 2013. Apply URL: <https://recruit.ucsc.edu/apply/JPF00016>

University of Dayton **Department Chair, Computer Science**

The Department of Computer Science at the University of Dayton invites applications for the position of Department Chair starting July 1, 2013. Applicants must have a Ph.D. in Computer Science and an academic record of achievement in teaching and scholarship appropriate for appointment with tenure at the rank of Associate Professor or Professor. Candidates must have a demonstrated ability to oversee curriculum development and innovation, communicate clearly and lead collaboratively, and have administrative experience at the department level. Preference will be given to those with experience in program assessment and accreditation and experience developing and/or teaching courses that cross department and unit boundaries. For a full description of the position and how to apply online see <http://jobs.udayton.edu/applicants/Central?quickFind=53763>.

A complete application consists of a cover letter, CV, graduate transcripts, and a personal statement addressing administrative experience, leadership style and teaching philosophy. In addition, the candidate should arrange for three



**THE UNIVERSITY of
NEW MEXICO**

Computer Engineering Faculty Position

The Department of Electrical and Computer Engineering (ECE) at the University of New Mexico invites applications for one or more full-time positions at the level of Assistant Professor. An Assistant Professor hire will be a probationary appointment leading to a tenure decision. Associate or Full Professor ranks will also be considered. Candidates must have earned a Ph.D., in Electrical or Computer Engineering or a closely related field by August 1st, 2013.

The preferred expertise areas are (i) bioengineering, (ii) hardware systems for Big Data, and (iii) computer networking and communications systems with an emphasis in applications such as biomedical image analysis and visualization, reconfigurable computing, and data analytics.

For complete job posting and how to apply, go to www.ece.unm.edu

The University of New Mexico is an equal opportunity/affirmative action employer and educator. We especially encourage members of underrepresented groups to apply.



NJIT
New Jersey's Science &
Technology University

Assistant/Associate/ Full Professor Faculty Positions Computer Science Department

New Jersey Institute of Technology (NJIT) invites applications for tenured/tenure-track positions beginning Fall 2013. The applications should work at the intersection of *Big Data with Cloud Computing* or *Wireless Networks* and have a PhD by Summer 2013 in Computer Science or related field. Department information: <http://cs.njit.edu>. To apply: upload cover letter, CV, research and teaching statements on njit.jobs (go to Search Postings and select Posting Number 0601259); Ask three (3) reference letter writers to email their letters to faculty-search@cs.njit.edu. **EOE/AA**

NEW JERSEY INSTITUTE OF TECHNOLOGY
UNIVERSITY HEIGHTS, NEWARK, NJ 07102-1982

THE EDGE IN KNOWLEDGE

letters of reference to be sent, by the closing date, to: Chair Search Committee, Department of Computer Science, University of Dayton, Dayton, OH 45469-2160. Applications and all related materials must be received by January 15, 2013. The search committee may be contacted at cps@udayton.edu.

The University of Dayton, founded in 1850 by the Society of Mary, is a Top Ten Catholic research university. The University seeks outstanding, diverse faculty and staff who value its mission and share its commitment to academic excellence in teaching, research and artistic creativity, the development of the whole person, and leadership and service in the local and global community. The University of Dayton received a National Science Foundation ADVANCE grant designed to promote the advancement and representation of women in STEM fields.

To attain its Catholic and Marianist mission, the University is committed to the principles of diversity, inclusion and affirmative action and to equal opportunity policies and practices. We act affirmatively to recruit and hire women, traditionally under-represented minority groups, people with disabilities and veterans.

University of Hawaii Dept of ICS Asst/Assoc/Full Professor

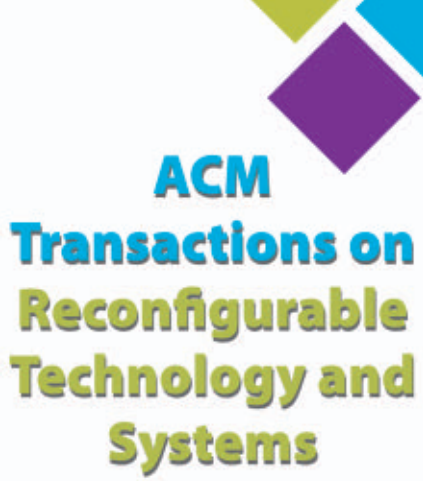
The Department of Information and Computer Sciences (ICS) at the University of Hawai'i at Manoa (UHM) is pleased to announce two tenure track faculty positions, pending availability of funds and position clearance. The ICS department at UHM has a more than 40-year history of excellence and innovation in information and computer science and offers 6 degrees: BA, BS, MS, MLISc and 2 PhDs.

Applicants must have a PhD from an accredited college or university in computer science or a closely related discipline. The ICS department is particularly interested in

applicants with a specialization in one of these areas: security/information assurance, big data analytics, computer graphics/visualization, or foundations of computing/algorithms. Salary and hiring rank will be commensurate with qualifications and experience. Applications should include a cover letter, vita, statement of research and teaching interests, and contact information for three references. Please send your application or any questions to cssearch@hawaii.edu or Search Committee Chair, Univ of Hawaii Dept of ICS, 1680 East West Rd, POST 317, Honolulu, Hawaii 96822. Additional information about the department can be found at www.ics.hawaii.edu. For the full job posting refer to workatuh.hawaii.edu, position #86062 & 82735 Closing Date: To receive full consideration, applications must be received by 1/18/2013. The University of Hawai'i is an Equal Opportunity Affirmative Action Employer.

University of Miami Tenure-track or Tenured Faculty Position

University of Miami invites applications and nominations for one open-rank, tenure-track or tenured faculty position in the department of Computer Science starting August 2013. Candidates must possess a Ph.D. in Computer Science or in a closely-related discipline with strong research experience in areas that pertain to collective and emergent behavior of highly connected systems such as social, biological, or dynamic networks. Candidates should also have experience in analyzing large data sets, as well as mathematical and computational modeling. Successful candidates will join a vigorous research program already underway across departments and schools within the university. The successful candidate will be expected to teach at both undergraduate and graduate levels and to develop and maintain internationally recognized research. Applicants should submit a cover letter, CV, research plan, statement of teaching philosophy,



ACM Transactions on Reconfigurable Technology and Systems



◆ ◆ ◆ ◆ ◆

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

◆ ◆ ◆ ◆ ◆

www.acm.org/trets
www.acm.org/subscribe



**Association for
Computing Machinery**





Inroads

Association for
Computing Machinery

Advancing Computing as a Science & Profession

http://inroads.acm.org

ACM Inroads

The magazine for
computing educators worldwide

Paving the way toward excellence in computing education

sample preprints or reprints, and the names of at least three references online to

<http://www.as.miami.edu/sciencecluster/>.

Review of applications will begin November 1, 2012 and continue until the positions are filled. Information about the College can be found at <http://www.as.miami.edu/> and about the department at <http://www.cs.miami.edu/>.

The University of Miami is an Affirmative Action/Equal Opportunity University that values diversity and has progressive work-life policies. Women, persons with disabilities, and members of other under-represented groups are encouraged to apply.

University of North Carolina Wilmington

Computer Science

Assistant Professor, Tenure-track

Vacancy 13F008 Starts August 2013. Ph.D. in Computer Science or closely related area required. Emphasis in computer graphics, visualization, animation or closely related area. **Details at** <http://uncw.edu/hr/employment-epa.html> **Priority consideration date: January 2, 2013.** EEO/AA Employer. Women and Minorities encouraged to apply.

University of Notre Dame Department of Computer Science and Engineering

Assistant or Associate Professor

The Department of Computer Science and Engineering at the University of Notre Dame invites applications for Assistant or Associate Professor. Excellent candidates in all areas will be considered.

The Department offers the PhD degree and undergraduate Computer Science and Computer Engineering degrees. Faculty are expected to excel in classroom teaching and to build and lead highly-visible research projects that attract substantial external funding.

The University of Notre Dame is a private, Catholic university with a doctoral research extensive Carnegie classification, and consistently ranks in USNWR as a top-twenty national university. The South Bend area has a vibrant and diverse economy with affordable housing and excellent school systems, and is within easy driving distance of Chicago and Lake Michigan.

Applicants should send (pdf format preferred) a CV, statement of teaching and research interests, and contact information for three professional references to: cse2013@nd.edu.

The University of Notre Dame is an Equal Opportunity, Affirmative Action Employer.

University of Puerto Rico at Mayaguez Assistant Professor

The Department of Mathematical Sciences at The University of Puerto Rico at Mayagüez invites applications for a tenure track position at the assistant professor level. It offers a Ph.D. Program in Computer and Information Science and Engineering (CISE) jointly with the Department of Electrical and Computer Engineering, an M.S. with options in Pure or Applied Mathematics or

Statistics, an M.S. in Scientific Computing, and three options leading to a B.S.: Pure Mathematics, Computer Science or Mathematics Education. The University of Puerto Rico at Mayaguez has a strong school of engineering and sciences, thus offering exceptional opportunities for interdisciplinary research and collaboration.

Qualifications include a Ph.D. in computer science, statistics, applied mathematics or a related area. It is expected that the candidate develop a strong record of research and teaching before tenure. Duties include undergraduate and graduate teaching and independent research. Initiative and leadership is expected. In addition, the applicant must submit a proposed model to bring external funding to support research.

To apply, send an application letter, AMS cover sheet, curriculum vitae, at least three letters of reference, and a work plan for the short, medium and long terms related to teaching and research. Deadline for applications is February 28th, 2013, however, applications will be received until the position is filled.

APPLY FOR THIS JOB

Apply URL: <http://math.uprm.edu>

Contact Person: Dr. Omar Colón Reyes

Email Address: omar.colon4@upr.edu

Phone: 787-265-3848

Fax: 787-265-5454

University of Rochester

Faculty Positions in Computer Science: HCI and Big Data

The University of Rochester Department of Computer Science seeks applicants for multiple tenure track positions in **human-computer interaction (HCI)** and **big data research** (including machine learning and data mining, cloud computing, e-science applications, and very large databases) Candidates must have a PhD in CS or a related discipline. Applicants for the big data position will also be considered for a faculty search in that area by our Department of Electrical & Computer Engineering. Additional information and online application instructions appear at <http://www.cs.rochester.edu/dept/recruit>.

The Department of Computer Science is a research-oriented department with a distinguished history of contributions in systems, theory, artificial intelligence, and HCI. We have a collaborative culture and strong ties to cognitive science, linguistics, and ECE. Over the past decade, a third of its PhD graduates have won tenure-track faculty positions, and its alumni include leaders at major research laboratories such as Google, Microsoft, and IBM.

The University of Rochester is a private, Tier I research institution located in western New York State. The University of Rochester consistently ranks among the top 30 institutions, both public and private, in federal funding for research and development. Teaching loads are light and classes are small. Half of its undergraduates go on to post-graduate or professional education. The university includes the Eastman School of Music, a premiere music conservatory, and the University of Rochester Medical Center, a major medical school, research center, and hospital system. The greater Rochester area is home to over a million people, including 80,000 students who attend its 8 colleges and universities.

The University of Rochester has a strong commitment to diversity and actively encourages applications from candidates from groups under-represented in higher education. The University is an Equal Opportunity Employer.

University of South Alabama Full-Time Tenure Track Faculty

The School of Computing at the University of South Alabama is soliciting applications for a full-time tenure-track faculty position in our Computer Science program beginning August 2013. Applicants at all ranks will be considered.

Applicants must have a Ph.D. in Computer Science or Computer Engineering with special consideration given to applicants with research interest in cyber security and assurance.

Details are available at <http://www.southalabama.edu/academicaffairs/facultyposition.html>. Review of applications will begin upon receipt and continue until the position is filled. The University of South Alabama is an Equal Opportunity/Equal Access Employer.

University of South Carolina Upstate Assistant Professor of Computer Science –

Requisition #004744

SEARCH EXTENDED -

NEW START DATE 8/16/2013

Teach courses in computer science. PhD required. For requirements and online application submission process, go to www.uscupstate.edu/jobs.

Contact: Dr. Jerome Lewis, Chair, jlewis@uscupstate.edu.

USC Upstate is an equal opportunity institution.

University of South Florida Faculty Position in Computer Security Computer Science and Engineering

Applications are invited for a tenured or tenure-track position at any rank in the Department of Computer Science and Engineering. The Department is seeking to hire in any area of computer security (cybersecurity), including system security, mobile- and medical-device security, anomaly detection, formal verification, usability, privacy, authentication, biometrics, machine learning and web security. Candidates should have an established record of sponsored research with outstanding-quality research publications and with potential for excellence in teaching. Candidates should also have a strong desire to provide leadership towards the establishment of a Center for Cybersecurity. Candidates must have completed a Ph.D.

The Department of Computer Science and Engineering (<http://www.cse.usf.edu>) has 24 faculty members and offers B.S., M.S., and Ph.D. degrees. The research program is well supported by federal and state agencies and industry. The University of South Florida serves over 47,000 students and is one of the nation's top public research universities.

For further information and for application instructions, please see our faculty search website: <http://www.cse.usf.edu/faculty-search/>. For questions please send email to faculty-search@

cse.usf.edu. Applications will be considered starting immediately until the position is filled.

According to Florida law, applications and meetings regarding them are open to the public. The University of South Florida is an Equal Opportunity/Equal Access/Affirmative Action Institution. Women and minorities are strongly encouraged to apply.

**University of Southern California
Marshall School of Business
Tenure Track Positions in Operations Management in the Marshall School of Business**

The Marshall School of Business invites applications for a tenure-track position in Operations Management starting in Fall 2013. Rank is open. The candidate should demonstrate a strong potential for scholarly research in Operations Management and an interest in teaching Operations Management at the undergraduate, graduate and doctoral levels. Candidates should have a Ph.D. in Operations Management (or a related discipline) or be assured of its completion by Fall 2013. Candidates with interdisciplinary interests in Operations Management and other areas such as Operations Research, Statistics, Information Systems, Economics, and Marketing are welcome. Special consideration will be given to candidates working in the emerging field of "Big Data".

Applicants should send a curriculum vita, copies of representative research papers or publications, evidence of teaching if any, and should arrange for three references letters to be sent to:

OM Search Committee
IOM Department, BRI-401
3670 Trousdale Parkway
Marshall School of Business
University of Southern California
Bridge Hall 401
Los Angeles, CA 90089-0809

Direct email inquiries to: Charlene.Conston@marshall.usc.edu

The applications should be received by December 1, 2012. Information about the IOM department, its faculty and the Marshall school can be obtained at: www.marshall.usc.edu/iom.

The University of Southern California values diversity and is committed to equal opportunity in employment. Women and men, and members of all racial and ethnic groups are encouraged to apply.

Located near the heart of Los Angeles, the University of Southern California is home to approximately 31,000 students and 4,300 faculty. USC Marshall is renowned for its high-ranking undergraduate, graduate, international and executive education programs, an exceptional faculty engaged in leading-edge research, a diverse and creative student body, and a commitment to technological advancement. The research productivity of Marshall's 200 full-time faculty ranks among the top 15 business schools in the world.

**University of Texas-Pan American
Computer Science Department
Computer Engineering Assistant Professor
Position**

The Department of Computer Science (<http://utpa.edu/cs>) at the University of Texas-Pan American (UTPA) seeks applications for a Tenure-Track

Assistant Professor position in Computer Engineering (**F12:13-013**). All candidates must have a potential/proven record in teaching and active research. The position requires a Ph.D. in computer science, computer engineering or related field. All areas of Computer Engineering will be considered. Of particular interest are candidates with teaching and research expertise in areas of Software Engineering, Computer Architecture, Gaming, and Computer Systems. The Computer Engineering program (<http://utpa.edu/cmpe>) is jointly administrated by the Computer Science and Electrical Engineering Departments and offers an ABET-accredited BS in Computer Engineering degree.

UTPA is situated in the lower Rio Grande Valley of South Texas, a strategic location at the center of social and economic change. With a population of over one million, the Rio Grande Valley is one of the fastest growing regions in the country. The region has a very affordable cost-of-living. UTPA is a leading educator of Hispanic/Latino students, with enrollment of over 19,000.

The position starts in Fall 2013, pending budget approval. The salary is competitive. A complete application should include: (1) a cover letter, specifically stating an interest in the Computer Engineering Assistant Professor position, noting your specialization, (2) vita, (3) statements of teaching and research interests, and (4) names and contact information of at least three references. Applications can be mailed to Dean's Office, Computer Science/Computer Engineering Search, College of Engineering and Computer Science, The University of Texas-Pan American, 1201 W. University Drive, Edinburg, Texas 78539 or emailed to coec@utpa.edu, ATTN: CS/CE Search. Review of materials will begin on November 1, 2012 and will continue until position is filled. Incomplete applications will not be considered. All UTPA employees are required to have a criminal background check (CBC). CBCs will be conducted on all candidates invited to campus.

UTPA is an Affirmative Action/Equal Opportunity employer. Women, minorities, and qualified individuals with disabilities are encouraged to apply. This position is security-sensitive as defined by the *Texas Education Code §51.215(c)* and *Texas Government Code §411.094(a)(2)*. Texas law requires faculty members whose primary language is not English to demonstrate proficiency in English as determined by a satisfactory paper-based test score of 500 (computer-based of 173 or internet-based of 61) on the Test of English as a Foreign Language (TOEFL) or a satisfactory test score of 6.0 on the International English Language Testing System (IELTS).

**The University of Texas at San Antonio
Faculty Positions in Computer Science**

The Department of Computer Science at **The University of Texas at San Antonio** invites applications for one or more tenure/tenure-track positions at the **Assistant, Associate or Professor** level, starting Fall 2013. We are particularly interested in candidates in software engineering, distributed systems, and operating systems. Other related areas of computer science will also be considered.

The Department of Computer Science currently has 22 faculty members and offers B.S., M.S., and Ph.D. degrees supporting a dynamic and growing

program with 634 undergraduates and more than 160 graduate students, including 81 Ph.D. students. See <http://www.cs.utsa.edu/fsearch> for **application instructions and additional information on the Department of Computer Science**.

Screening of applications will begin on January 2, 2013 and will continue until the positions are filled or the search is closed. UTPA is an EO/AA Employer.

**Chair of Faculty Search Committee
Department of Computer Science
The University of Texas at San Antonio
One UTSA Circle
San Antonio, TX 78249-0667
Phone: 210-458-4436
Fax: 210-458-4437**

**University of Wisconsin-Stevens Point
Assistant/Associate Professor
Teach Computing and New Media
in Central Wisconsin**

The **University of Wisconsin-Stevens Point** is accepting applications for a tenure track position in CNMT for Aug 2013. Doctoral degree and experience required. Web-based application development and design with specialty areas will include one or more of: user experience analysis and design, object-oriented programming, Web/mobile development, and a solid understanding of the software development life cycle. Application processing will begin on January 31, 2013, and continue until the position is filled. For additional information contact Prof Tim Krause: tkrause@uwsp.edu, 715-346-3851; or see our Web sites: www.uwsp.edu/equity and www.uwsp.edu/cnmt
AA/EOE

**Washington University in St. Louis
Faculty Positions**

The Department of Computer Science & Engineering at Washington University in St. Louis seeks outstanding tenure-track faculty in all areas of computer science and engineering at the assistant professor level. Exceptional candidates at the associate and full professor levels will also be considered. The department plans to grow its faculty size by 50% in the coming years. We seek multiple talented and highly motivated individuals who will build transformative research programs, both through work in the core disciplines of computer science and computer engineering and through interdisciplinary collaborations with researchers in areas such as biomedicine, engineering, and the sciences. Successful candidates must show exceptional promise for research leadership and a strong commitment to high-quality teaching. Candidates will be expected to publish their research in peer-reviewed journals, to teach, and to participate in department and University service. For full information about this search and application instructions, please visit: <http://cse.wustl.edu/aboutthedepartment/Pages/OpenFacultyPositions.aspx>. Applicants should hold a doctorate in Computer Science, Computer Engineering, or a closely related field. Washington University in St. Louis is an Equal Opportunity and Affirmative Action employer and invites applications from all qualified candidates. Employment eligibility verification required upon employment.



DOI:10.1145/2380656.2380680

Peter Winkler

Puzzled

Solutions and Sources

Last month (November 2012) we posted a trio of brainteasers concerning the use of a balance scale to determine the weight of various numbers of coins. Here, we offer solutions to all three. How did you do?

1. Same weight.

This lovely puzzle was from a Russian competition and included in *The USSR Problem Book* by D.O. Shklarsky et al., W.H. Freeman and Co., San Francisco, 1962. We wanted to show that if 13 coins have the property that any 12 of them can be divided into two stacks of six coins each that balance on the scale, then all the coins would have the same weight.

Now suppose the weights are not all the same. If the weights are all integers, we can reach the following contradiction: Shift the weights down (does not affect either the weighing property or the conclusion) so the lightest coin has weight 0. Now keep dividing all the weights by two until there is a coin of odd weight. If we leave the odd-weight coin behind, the sum of the weights of the other coins must be even, since we can balance them. However, the same must be true if we leave the weightless coin behind, which is impossible.

This argument also works if all the weights are rational numbers, since we can change units so the weights are integers. If the weights are irrational, we would be tempted to replace each

weight with a nearby rational number, then proceed as above. Annoyingly, however, the contradiction we arrived at earlier does not materialize in the presence of approximations. Instead, we express the real numbers as a vector space over the rationals; the rest we leave to the intrepid reader.

2. Three weighings.

This puzzle was brought to my attention by Noga Alon of Tel Aviv University, and much more about it can be found in the article “Coins and Cones” by Dmitry Kozlov and Van Vu in the *Journal of Combinatorial Theory (Series A)* 78, 1 (1997), 1–14. The problem was to determine whether, in three weighings, eight coins with at most two different weights actually all weigh the same. We can handle 2^n coins in n steps, in particular eight coins in three steps, by first dividing the coins into two equal stacks; assuming they balance, now divide one of the stacks into two equal stacks and weigh them, then repeat. If there are coins of two weights, then one of these weighings must fail to balance.

3. Vectors that sum to zero.

For years mathematicians thought the algorithm just outlined is optimal; that is, that no more than 2^n coins can be handled by n weighings. Kozlov and Vu found a beautiful way to look at the problem using sets of vectors that sum to zero; for other applications of such sets, see the “Puzzled” column “Find the Magic Set” (Aug. 2012). Among other things, their work uncovered counterexamples to the previous view; in particular, the following way to do 10 coins in three weighings: 1,2,3,4,5 vs. 6,7,8,9,10; 1,2,3,4 vs. 5,6,7,8; and 6,7,8 vs. 5,9,10.

Give yourself a pat on the back if you solved this puzzle, with or without mathematical machinery.

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.

© 2012 ACM 0001-0782/12/12

[CONTINUED FROM P. 128] so the salesman is allowed to cross back and forth between the subsquares to a limited extent. Formally, this technique is dynamic programming, which is very old. Then I showed that in order to come within Epsilon of the optimum solution, you only need to allow something like 10 over Epsilon crossings. So that was the idea: If you allow the salesman to go between these smaller squares a few times, you can still get a near-optimum solution, and there's a simple dynamic programming language that enables you to compute the tour with these restrictions.

You've also worked on the use of semidefinite programming, an extension of linear programming that was developed in the 1980s, in approximation algorithms.

Semidefinite programming gives a very general methodology for computing approximate solutions to NP-complete problems. First, you try to relax the problem—meaning that you create an intuitive approximation by throwing away some features—such that it can be represented by a linear or semidefinite program. Then you solve that program.

So semidefinite programming can be used to find approximations for many NP-complete problems.

Yes, but the question is, How much do you lose when you relax the problem? For many problems, people can quickly come up with instances where you lose too much. For others, like graph partitioning, it's hard to find counterexamples. That was the status of semidefinite programs; there were these methods to approximate various problems, and we didn't know how well they worked for many problems. So while Umesh Vazirani, Satish Rao, and I were looking at graph partitioning, we came up with a powerful way to upper bound the amount of approximation, which other people went on to use for all kinds of things. Then in some follow-up work with my students I looked at how to solve semidefinite programs themselves and showed how to solve them much more efficiently.

You've also been involved with work on the Unique Games Conjecture (UGC), a deceptively simple proposal formu-

lated by your former student Subhash Khot about the difficulty of assigning colors to the vertices of a network.

The UGC has a very interesting connection to semidefinite programming because it seems to imply that for many problems, semidefinite programming is the best way to approximate the problem. So, in fact, it can be shown that if the semidefinite program doesn't work, then no other technique works. My research is just a small part of that; many other people were involved.

Your recent work with Boaz Barak and David Steurer proved the existence of a subexponential algorithm for the UGC. What are the implications of that discovery?

It doesn't quite resolve the UGC, but it certainly puts it in a very new light. We don't know of other NP-complete problems that have subexponential algorithms, so if the Unique Games problem is NP-complete, as is claimed by the UGC, then it is a fairly atypical problem. Moreover, many experts think the subexponential algorithm may be improvable further to disprove the UGC.

Another thing you have done to help solve difficult problems was cofound Princeton's Center for Computational Intractability.

I'm very interested in this broader issue of computational intractability because it has all kinds of implications. A lot of problems in artificial intelligence, at least the way they are phrased, are intractable. What does that imply either for artificial intelligence or for our understanding of the brain? If certain physics problems are intractable, what does that say about physics? Four member institutions are involved in the center, and we hold workshops and meetings.

Can you talk about some of the research that has come out of your work with the center?

The above work on UGC is one. Another interesting thing I got into was the implication of intractability for finance. This was motivated by all the talk about financial derivatives and their role in the financial crash of 2008. When I started looking at derivatives, I was struck by the fact that all the computational problems that were connected with them

seemed to be intractable problems. So some colleagues and I started discussing it. In economic theory, it turns out, there are various justifications for derivatives. One is that they help alleviate the problem of asynchronous information.

This is the so-called lemon problem of George Akerlof, which addresses the fact that in a market, some people have more information than others.

And the idea is that because of this, markets shouldn't work, because people who think they have less information will be too scared to enter into transactions. In practice, there are many ways to alleviate the lemon problem. Economic theory suggests derivatives mitigate it. What we showed is that because of computational intractability, there's something wrong with that theoretical justification. Thus far some experts had said, "OK, there were problems with how derivatives were used in real life, but in theory they are fine." Our work raises questions about this.

How?

The old work justifying derivatives looked at a simplified case where there's just one seller who is selling one derivative. This is not atypical in economic theory, where you study issues in simplified models. In real life, buyers have to sift through a lot of financial products in the marketplace, and that's where you have to watch out for intractability. We show that computational intractability arises even when a single seller is selling many derivatives.

I imagine there are a lot of other settings that intractability could shed light on.

Our center is working on that, too. One issue that interests me is the intractability of data analysis problems. We know intuitively that sifting through large amounts of data is a tough problem, but computational intractability can help us formalize it. But I am interested in understanding whether or not intractability is actually a barrier. I suspect that in many cases the intractability can be bypassed by improved modeling. 

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

© 2012 ACM 0001-0782/12/12

DOI:10.1145/2380656.2380681

Leah Hoffmann

Q&A

As Good As It Gets

Sanjeev Arora talks about proof, intractability, and finding the best way to approximate problems.

SANJEEV ARORA HAS long been drawn to his field's most intractable challenges. Born in Rajasthan, India, the Princeton University professor is best known for his work on the development of probabilistically checkable proofs and a proof of the PCP theorem, which has helped distinguish between problems that can and cannot be approximated efficiently. He has also contributed new approximation algorithms for classic problems like the Euclidean traveling salesman problem and graph partitioning. More recently, he has cast new light on the Unique Games Conjecture, a problem that goes beyond the PCP theorem to imply that the known approximation algorithms for many simple problems are the best possible.

Let's start with your work on the PCP theorem, which, if I understand correctly, states that for anything you can prove, you can write the proof in such a way that it can be verified by looking at only a few bits.

That's the basic idea. The concept was beginning to be developed just before I started grad school. I started working on it with some collaborators, and that became my dissertation.

You uncovered several key insights. Can you explain them?

One was the idea that we could improve the efficiency of verification by using a kind of circular reasoning. For instance, if I'm writing a proof that somebody else is going to verify by looking at just a few bits, then the statement that the verifier will accept the proof is itself something that can



be proved using the same technique. It's a smaller statement, and you can use the same ideas to give a proof for that. Of course, this is the usual computer science notion of "recursion" and it increases the efficiency. The overall procedure is algebraic. You don't normally think of verifying proofs as an algebraic procedure, but we were able to express it algebraically. This is actually quite common in mathematics—you're trying to prove something about, say, knots or strings and you encode it as an algebraic question, and from then on you can use tools from algebra.

You also introduced a new definition of NP.

It was an extension of NP-completeness that allowed people to prove that for certain NP-complete problems, it is NP-complete even to compute approximate solutions. In other words, comput-

ing an approximate solution is no easier than computing the optimum solution. The PCP theorem itself doesn't tell you the problems for which it's impossible to calculate approximate solutions. But it provided the starting point for other results that do, and many such results have been proved in the subsequent two decades. And after the full body of work, we have been able to chart the landscape of those problems. For many problems, we have even determined the exact threshold of approximability: We have algorithms that approximate it up to some factor, and we know that doing any better is NP-complete. Of course, such results also sometimes involved designing new approximation algorithms for various problems.

You helped develop some of those approximation algorithms yourself, including one for the Euclidean traveling salesman problem (TSP), a variation of the classic NP-complete problem in which a salesman is trying to find the shortest path by which he can visit all of his N clients.

As I thought about the PCP theorem, I also began wondering about whether it was hard to compute approximate solutions to the Euclidean TSP, which is a classic problem. And I slowly started seeing structure in the problem that nobody else had until that point, which led to the algorithm.

The idea sounds simple: Divide and conquer.

It's something that people had tried before and had given up on. Say you have a square that encloses all the points, and you imagine dividing the square into subsquares. And somehow you compute a reasonable tour within each subsquare, and then you put them all together. I put a twist on that idea. In the classic divide-and-conquer scheme, you solve each subsquare independently. My technique was to allow a little bit of dependence, [CONTINUED ON P. 127]



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

Computing Reviews is on the move



Our new URL is

ComputingReviews.com



COMPUTING REVIEWS

A daily snapshot of what is new and hot in computing