

COMMUNICATIONS

CACM.ACM.ORG

OF THE

ACM

09/08 VOL.51 NO.9

Enterprise Information Integration and other tools for merging data

How Do I
Model State?

Formal Methods

Design and Code
Reviews In the
Internet Age

Power Management

The Puzzle of Apple

Science Policy Isn't
Always About Science



CSCW08

November 8-12, 2008
Hilton San Diego Resort
San Diego, California, USA

CALL FOR REGISTRATION

The 2008 ACM Conference
on Computer Supported
Cooperative Work

Attend CSCW 2008 for an intellectually stimulating and diverse technical program!

Plenary Speakers

The CSCW 2008 Opening and Closing Keynote speakers cover the spectrum of interests across the CSCW community from the technical and design challenges of creating collaboration systems to the intricacies of their impact on people and practices. The opening Keynote will be by Cory Ondrejka, co-founder of Second Life, and the Closing Keynote will be given by Sara Diamond, president of the Ontario College of Art & Design.

Paper Sessions

The paper sessions will cover emerging areas of collaboration and collaborative systems such as Wikis & Wikipedia, Naughty & Nice issues in Social Networking Systems, Health Informatics, Deployments at Home, and Disrupted Environments. The paper sessions will also include traditional CSCW topics such as Distributed Teams, Media Spaces, Community, Interpersonal Relations in the Group, Work Places and Work Practices.

Workshops

Please consider participating in a CSCW 2008 Workshop! There are more than 10 workshops scheduled during the weekend prior to the main technical program. Some of the scheduled workshops include:

- *Social Networking in Organizations*
- *The Future of Mobile Social Software*
- *Virtual Radical Collocation*
- *Remix rooms: Mashing up media for meetings CSCW and Human Factors – Where are we now and what are the challenges?*
- *Changing work, Changing technology (IFIP 9.1 WG)*

Key Registration Dates

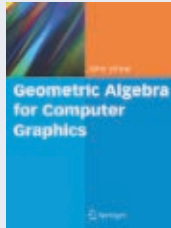
Aug 15, 2008 - Registration Opens | Sept 30, 2008 - Early Registration Deadline | Oct 3, 2008 - Hotel Room Reservation Deadline

CSCW 2008 also contains Panels, Demonstrations and Interactive Poster sessions that provide opportunities to engage in lively discussions with leaders in this field of increasing importance to our global future.

Questions and requests for information should be sent to publicity@cscw2008.org, or visit our website: www.cscw2008.org

W W W . C S C W 2 0 0 8 . O R G

Computer Science Textbooks



Geometric Algebra for Computer Graphics

J.A. Vince

Since its invention, geometric algebra has been applied to various

branches of physics such as cosmology and electrodynamics, and is now being embraced by the computer graphics community where it is providing new ways of solving geometric problems. It took over two thousand years to discover this algebra, which uses a simple and consistent notation to describe vectors and their products. Vince tackles this new subject in his usual inimitable style, and provides an accessible and very readable introduction.

2008. XVI, 256 p. 125 illus. Hardcover
ISBN 978-1-84628-996-5 ► **\$99.00**

Exploring Computer Science with Scheme

O. Grillmeyer

The aim of this textbook is to present the central and basic concepts, techniques, and tools of computer science. The emphasis is on presenting a problem-solving approach and on providing a survey of all of the most important topics covered in computer science degree programmes. Scheme is used throughout as the programming language and the author stresses a functional programming approach which concentrates on the creation of simple functions that are composed to obtain the desired programming goal. Such simple functions are easily tested individually.

2nd ed. 2009. Approx. 610 p. (Undergraduate Texts in Computer Science) Hardcover
ISBN 978-0-387-76624-9 ► **\$79.95**



Computational Geometry Algorithms and Applications

Algorithms and Applications

M.d. Berg, O. Cheong,
M.v. Kreveld,
M. Overmars

This well-accepted

introduction to computational geometry is a textbook for high-level undergraduate and low-level graduate courses. The focus is on algorithms and hence the book is well suited for students in computer science and engineering. In this third edition, besides revisions to the second edition, new sections discussing Voronoi diagrams of line segments, farthest-point Voronoi diagrams, and realistic input models have been added

3rd ed. 2008. XII, 386 p. 370 illus. Hardcover
ISBN 978-3-540-77973-5 ► **\$49.95**



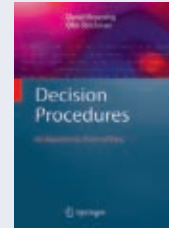
Bioinformatics Problem Solving Paradigms

V. Sperschneider

There are fundamental principles for problem analysis and algorithm design that are

continuously used in bioinformatics. This book concentrates on a clear presentation of these principles, presenting them in a self-contained, mathematically clear and precise manner. This book highlights basic paradigms of problem analysis and algorithm design in the context of core bioinformatics problems. Mathematically demanding themes are put across to the reader by properly chosen representations with the aid of lots of illustrations.

2008. CCLXXXIX, 18 p. 208 illus. Hardcover
► **With contributions by Jana Sperschneider and Lena Scheubert.**
ISBN 978-3-540-78505-7 ► **\$59.95**



Decision Procedures

An Algorithmic Point of View

D. Kroening,
O. Strichman

The book concentrates on decision procedures

for first-order theories that are commonly used in automated verification and reasoning, theorem-proving, compiler optimization and operations research. The techniques described in the book draw from fields such as graph theory and logic, and are routinely used in industry.

2008. XVI, 304 p. 67 illus. (Texts in Theoretical Computer Science. An EATCS Series) Hardcover
ISBN 978-3-540-74104-6 ► **\$69.95**



The Algorithm Design Manual

S.S. Skiena

This newly expanded and updated second edition of the best-selling classic continues to take the "mystery" out

of designing algorithms, and analyzing their efficacy and efficiency. Expanding on the first edition, the book now serves as the primary textbook of choice for algorithm design courses while maintaining its status as the premier practical reference guide to algorithms for programmers, researchers, and students

2nd ed. 2008. 115 illus. With online files/update., Hardcover
ISBN 978-1-84800-069-8 ► **\$79.95**

Departments

- 5 **President's Letter**
ACM's Place in the Global Picture
By Wendy Hall
-
- 7 **Publisher's Corner**
50 Years Young
By Scott E. Delman
-
- 9 **Letters To The Editor**
Knuth's Art of Recovering from Errors
-
- 10 **CACM Online**
We're All Ears
By David Roman

100 **Careers**

News

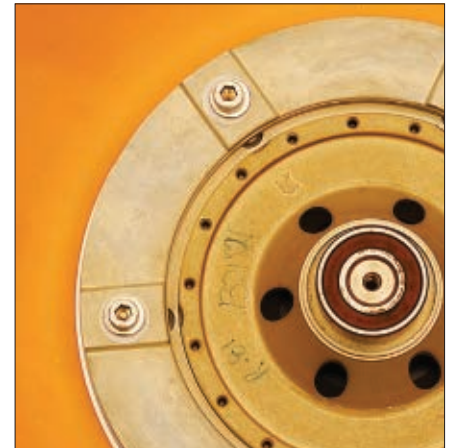
- 11 **Finding Diamonds in the Rough**
Spectral graph theory has proven to be very useful for text search and retrieval and for refining predictive-analysis systems.
By Kirk L. Kroeker
-
- 14 **Ubiquitous Video**
Scalable and distributed video coding offers the promise of two-way, real-time video.
By Logan Kugler
-
- 17 **Privacy Matters**
As concerns about protecting personal data increase, differential privacy offers a promising solution.
By Samuel Greengard
-
- 19 **Wisdom From Randy Pausch**
Weeks before his death on July 25, Randy Pausch graciously commented on his life's work, his hopes for computer science, and his regard for the students who are its future.
By Leah Hoffmann

Viewpoints



- 22 **Technology Strategy and Management**
The Puzzle of Apple
Given Apple's unique characteristics, should it strive to be a platform or a product leader?
By Michael Cusumano
-
- 25 **Kode Vicious**
Pride and Prejudice (The Vasa)
Navigating the well-traveled course of communication failure that often leads to engineering disasters.
By George Neville-Neil
-
- 27 **IT Policy**
Science Policy Isn't Always About Science
What is the appropriate role and level of influence for science and technical advice in policy deliberations?
By Cameron Wilson and Peter Harsha
-
- 30 **Viewpoint**
Global Warming Toward Open Educational Resources
Seeking to realize the potential for significantly improving and advancing the world's standard of education.
By Richard G. Baraniuk and C. Sidney Burrus

Practice



- 34 **How Do I Model State? Let Me Count the Ways**
A study of the technology and sociology of Web service specifications.
by Ian Foster, Savas Parastatidis, Paul Watson, and Mark McKeown
-
- 42 **Powering Down**
Smart power management is all about doing more with the resources we have.
By Matthew Garrett
-
- 47 **CTO Storage Roundtable, Part Two**
Leaders in the storage industry ponder upcoming technologies and trends.
By Mache Creeger, Moderator
-
- 54 **Software Engineering and Formal Methods**
The answer to software reliability concerns may lie in formal methods.
By Mike Hinchey, Michael Jackson, Patrick Cousot, Byron Cook, Jonathan P. Bowen, and Tiziana Margaria

Contributed Articles



60 **Beyond Keywords: Automated Question Answering on the Web**
Beyond Google, emerging question-answering systems respond to natural-language queries.
By Dmitri Roussinov, Weiguo Fan, and José Robles-Flores

66 **Design and Code Reviews in the Age of the Internet**
New collaboration tools allow geographically distributed software-development teams to boost the venerable concept of code review.
By Bertrand Meyer

Review Articles

72 **Information Integration in the Enterprise**
A guide to the tools and core technologies for merging information from disparate sources.
By Philip A. Bernstein and Laura M. Haas

About the Cover: An open source Processing application developed by computer scientist turned artist Leander Herzog generated the graphic for this month's cover and story opener on pages 72–73. Herzog notes working with code offers him a way to explore visual ideas very quickly, as the code breaks the relation between complexity, precision, and time, which often limit the design process.

Research Highlights

82 **Technical Perspective**
Transactional Memory in the Operating System
By Mark Moir

83 **TxLinux and MetaTM: Transactional Memory and the Operating System**
By Christopher J. Rossbach, Hany E. Ramadan, Owen S. Hofmann, Donald E. Porter, Aditya Bhandari, and Emmett Witchel

92 **Technical Perspective**
Distributing Your Data and Having It, Too
By Hagit Attiya

93 **Distributed Selection: A Missing Piece of Data Aggregation**
By Fabian Kuhn, Thomas Locher, and Roger Wattenhofer

Last Byte

103 **Puzzled**
Solutions and Sources
By Peter Winkler

104 **Future Tense**
Will
Expect virtual immortality through enduring, realistic avatars based on published work and archived memory.
By William Sims Bainbridge

Virtual Extension

As with all magazines, page limitations often prevent the publication of articles that might otherwise be included in the print edition. For this reason and to ensure the timely publication of high-quality articles, ACM created *Communications'* Virtual Extension (VE).

VE articles undergo the same rigorous review process as those in the print edition and are accepted for publication on their merit. In addition, VE articles are listed in the Table of Contents of the print magazine and are paginated and fully citable as is every article published in *Communications*. VE articles are published exclusively in the ACM Digital Library.

The following articles are an extension of the September 2008 print edition, now available to ACM members in the Digital Library.

Using and Fixing Biased Rating Schemes
By Robin Poston

Using Traceability to Mitigate Cognitive Biases in Software Development
By Kannan Mohan and Radhika Jain

Understanding User Perspectives on Biometric Technology
By Alexander P. Pons and Peter Polak

Following Linguistic Footprints: Automatic Deception Detection in Online Communication
By Lina Zhou and Dongsong Zhang

Toward Agility in Design in Global Component-Based Development
By Julia Kotlarsky, Ilan Oshri, Kuldeep Kumar, and Jos van Hillegersberg

The Student Productivity Paradox: Technology-Mediated Learning in Schools
By Neset Hikmet, Eileen Z. Taylor, and Christopher J. Davis

What Factors Drive the Assimilation of Internet Technologies in China?
By Patrick Y.K. Chau, Fujun Lai, and Dahui Li.



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 Membership
Lillian Israel
Director, Office of Publications
Mark Mandelbaum
Director, Office of SIG Services
Donna Cappo

ACM COUNCIL
President
Wendy Hall
Vice-President
Alain Chenais
Secretary/Treasurer
Barbara Ryder
Past President
Stuart I. Feldman
Chair, SGB Board
Alexander Wolf
Co-Chairs, Publications Board
Ronald Boisvert, Holly Rushmeier
Members-at-Large
Carlo Ghezzi;
Anthony Joseph;
Mathai Joseph;
Kelly Lyons;
Bruce Maggs;
Mary Lou Soffa;
SGB Council Representatives
Norman Jouppi;
Robert A. Walker;
Jack Davidson

PUBLICATIONS BOARD
Co-Chairs
Ronald F. Boisvert and Holly Rushmeier
Board Members
Gul Agha; Michel Beaudouin-Lafon;
Jack Davidson; Carol Hutchins;
Ee-ping Lim; M. Tamer Ozsu; Vincent Shen;
Mary Lou Soffa; Ricardo Baeza-Yates

ACM U.S. Public Policy Office
Cameron Wilson, Director
1100 Seventeenth St., NW, Suite 507
Washington, DC 20036 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Chris Stephenson
Executive Director
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (800) 401-1799; F (541) 687-1840

Association for Computing Machinery (ACM)
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
T (212) 869-7440; F (212) 869-0481

COMMUNICATIONS OF THE ACM

A monthly publication of ACM Media

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

GROUP PUBLISHER
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
Alicia Kubista
Assistant Art Director
Mia Angelica Balaquiot
Production Manager
Lynn D'Addesio
Director of Media Sales
Jonathan Just
Advertising Coordinator
Graciela Jacome
Marketing & Communications Manager
Brian Hebert
Public Relations Coordinator
Virginia Gold
Publications Assistant
Emily Eng

Columnists
Alok Aggarwal; Phillip G. Armour
Martin-Campbell-Kelly;
Michael Cusumano; Peter J. Denning;
Shane Greenstein; Mark Guzdial;
Peter Harsha; Leah Hoffmann;
Deborah Johnson; 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
acmcoa@cacm.acm.org
Letters to the Editor
letters@cacm.acm.org

WEB SITE
http://cacm.acm.org

AUTHOR GUIDELINES
http://cacm.acm.org/guidelines

ADVERTISING

ACM ADVERTISING DEPARTMENT
2 Penn Plaza, Suite 701, New York, NY
10121-0701
T (212) 869-7440
F (212) 869-0481

Director of Media Sales
Jonathan M. Just
jonathan.just@acm.org

Media Kit
acmmediasales@acm.org

EDITORIAL BOARD

EDITOR-IN-CHIEF
Moshe Y. Vardi
eic@cacm.acm.org

NEWS
Co-chairs
Marc Najork and Prabhakar Raghavan
Board Members
Brian Bershad; Hsiao-Wuen Hon;
Mei Kobayashi; Rajeev Rastogi;
Jeannette Wing

VIEWPOINTS
Co-chairs
William Aspray;
Susanne E. Hambrusch;
J Strother Moore
Board Members
Stefan Bechtold; Judith Bishop;
Peter van den Besselaar; Soumitra Dutta;
Peter Freeman; Seymour Goodman;
Shane Greenstein; Mark Guzdial;
Richard Heeks; Susan Landau;
Carlos Jose Pereira de Lucena;
Helen Nissenbaum; Beng Chin Ooi

PRACTICE
Chair
Stephen Bourne
Board Members
Eric Allman; Charles Beeler;
David J. Brown; Bryan Cantrill;
Terry Coatta; Mark Compton;
Ben Fried; Pat Hanrahan;
Marshall Kirk McKusick;
George Neville-Neil

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

CONTRIBUTED ARTICLES
Co-chairs
Al Aho and George Gottlob
Board Members
Yannis Bakos; Gilles Brassard; Peter Buneman; Andrew Chien; Anja Feldman;
Blake Ives; Takeo Kanade; James Larus;
Igor Markov; Gail C. Murphy; Shree Nayar;
Lionel M. Ni; Sriram Rajamani; Avi Rubin;
Abigail Sellen; Ron Shamir; Larry Snyder;
Wolfgang Wahlster; Andy Chi-Chih Yao;
Willy Zwaenepoel

RESEARCH HIGHLIGHTS
Co-chairs
David A. Patterson and Stuart J. Russell
Board Members
Martin Abadi; P. Anandan; Stuart K. Card;
Deborah Estrin; Stuart I. Feldman;
Shafi Goldwasser; Maurice Herlihy;
Norm Jouppi; Andrew B. Kahng; Linda Petzold; Michael Reiter;
Mendel Rosenblum; Ronitt Rubinfeld;
David Salesin; Lawrence K. Saul;
Guy Steele, Jr.; Gerhard Weikum

WEB
Co-chairs
Marti Hearst and James Landay
Board Members
Jason I. Hong; Jeff Johnson;
Wendy MacKay; Jian Wang

ACM Copyright Notice

Copyright © 2008 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

Annual subscription cost is included in the society member dues of \$99.00 (for students, cost is included in \$42.00 dues); the nonmember annual subscription rate is \$100.00.

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://cacm.acm.org/advertising> or by contacting ACM Media Sales at (212) 626-0654.

Single Copies

Single copies of *Communications of the ACM* are available for purchase. Please contact acmhelp@acm.org.

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM* 2 Penn Plaza, Suite 701 New York, NY 10121-0701 USA



Association for Computing Machinery

Printed in the U.S.A.



DOI:10.1145/1378727.1378728

Wendy Hall

ACM's Place in the Global Picture

ACM's new president intends to make international initiatives a top priority, hoping to share the Association's riches with a greater global audience.

It is a tremendous honor to have been elected ACM President. My history with this Association goes back many years and in that time I've witnessed great advances in our profession and continue to be impressed by ACM's leadership role in responding to those changes with premier publications, conferences, curricula models, and professional services that reflect emerging computing research.

I recall my first ACM invitation to join its Publications Board, never imagining my acceptance of that invitation would one day lead to my becoming President. Back then I looked forward to those board meetings not only for the impact we were making in bringing great new journals to the computing field but for the chance to visit ACM headquarters in New York, one of my favorite cities. But that was over 20 years ago, when you still had to walk to the library to find your favorite journal and attending conferences or buying conference proceedings was the most effective way to keep up with your area of expertise. In fact, two of the major attractions for me becoming an ACM member was the value such membership gave a computer science researcher at that time—a more economical means for subscribing to top journals and attending professional conferences.

Today we live in a post-Web world where almost everything we want to read is available online as soon as it is published. Indeed, the ups and downs of the major world economies have affected many of us to the point where

traveling to our favorite conferences is sometimes viewed as a luxury rather than a basic necessity.

With all its publications available through the Digital Library and an increasing number of conferences being held outside the U.S., the location of ACM's headquarters is no longer viewed by its physical address. ACM is everywhere its members are; and that location continues to expand globally. ACM membership numbers are in fact steadily increasing. It is my ambition while I am President to ensure this trend continues and to set in motion initiatives to accelerate it and to increase the diversity of membership in the widest sense.

There is work to be done. It remains a frustration that many people from outside the U.S. think the 'A' in ACM stands for American. Our goal must be to find ways to spread the word about ACM and to illustrate how our vast array of valued resources and professional services have relevance to every computing professional, regardless of location.

Certainly the recent revitalization of *Communications* is a crucial element in the process of increasing our international presence and expanding the service we provide to all of our communities. We are also continuing to develop our services to the practitioner community by making *ACM Queue* available online as part of a new *Queue* Web site, which will include many features and content channels specifically targeted to practitioners. You will also find in *Communications* a new "Practice" sec-

tion that covers the issues and technology trends facing today's practitioners.

We plan to develop further our initiatives in India and China, explore our relationship with Europe, and examine how to position our services and publications to be more relevant in South America and other parts of the world. We also intend to give a higher profile to ACM-W in order to make the Association more pertinent to the women in our community and to encourage more women to consider careers in computing.

I look forward to a very exciting and fruitful two years as President and hope to meet as many of you as possible at various ACM events and conferences during that time. Every ACM President wants to make a difference, and as its first non-North American President I hope when I look back on my term in office I will be able to see demonstrable evidence that the ACM has increased its relevance and attractiveness as a membership organization to the worldwide computing community.

Wendy Hall (wh@ecs.soton.ac.uk) is president of the ACM and a professor of computer science at the University of Southampton, U.K.

The 1st ACM SIGGRAPH Conference and Exhibition in Asia
www.siggraph.org/asia2008

New Horizons

ACM SIGGRAPH launches the premiere SIGGRAPH Asia in Singapore

Programs include:

- Papers
- Sketches and Posters
- Courses
- Art Gallery
- Computer Animation Festival
- Educators Program
- Emerging Technologies
- Exhibition

***Calling all creative researchers, artists,
digital innovators and animators!***

This is your opportunity to present your stellar work or attend the 1st ACM SIGGRAPH conference and Exhibition in Asia.

Queries?

Contact Conference and Exhibitions
Management at asia2008@siggraph.org or
Tel: +65 6500 6700



SIGGRAPHASIA2008

Conference and Exhibition on Computer Graphics and Interactive Techniques
Singapore, 10-13 December 2008

Held in
UNIQUELY
Singapore
visitsingapore.com



DOI:10.1145/1378727.1378729

Scott E. Delman

“

At the end of the day this is a magazine by and for the computing community, and like any other community initiative is almost entirely dependent upon the participation of its members.

”

50 Years Young

I've received dozens of email messages providing feedback on the July issue.

More than any other sentiment expressed in those messages is the sense of excitement that *Communications* has taken a positive step forward and at the same time has returned to its roots. As Publisher, I am both gratified and conflicted by this feedback. In many ways, *Communications* is an icon for the computing community, with a long-standing tradition of quality that has stood the test of time. Changing an icon is not an easy thing to do and is not without significant risks, especially when the icon is also the flagship publication for the community's leading Association and as such the primary vehicle for communicating what is happening across the computing field. Nevertheless, this change was necessary because both the community and field have been evolving and growing so rapidly in recent years that a new and more comprehensive global voice needed to be created simply to keep pace.

This *Communications* is more than a revamped version of a 50-year-old publication, it is a completely new magazine with an entirely new voice. With a new look and feel, a new editorial board, a new editorial scope, and a more global vision, *Communications* is attempting to appeal to a broader cross section of the computing field and become even more relevant for those across the entire discipline, as well as those entering the field and those working on its fringes. An example of this is the decision to expand the News section and hire professional journalists to cover a wider range of timely and important topics. In each issue, you can expect to find three in-depth news articles, one written from a technology perspective, one from a science perspective, and one from a so-

cietal perspective. The depth of these articles is greater than can be found in less specialized publications, in keeping with *Communications'* emphasis on technical sophistication and rigor.

Throughout the magazine, you will find other examples of ways *Communications* is reaching out to a broader community. These include the creation of the Practice section, aimed primarily at those working in industry but appealing to all *Communications* readers. Another is the introduction of Technical Perspectives that accompany full-length research papers and intended to make pure computer science research more accessible and contextually relevant to the practitioner and educator.

While the verdict is far from in, I am pleased to say that I received a number of letters from loyal readers who informed me the July issue was one of the first they've read cover to cover in a number of years, but at the same time they were skeptical that it would be possible to keep the magazine as relevant on a consistent basis. My response to this is simple. The magazine has a new editorial board second to none, an incredible publishing staff committed to making *Communications* better with every issue, and an organization behind it that supports all of this effort. But at the end of the day this is a magazine by and for the computing community, and like any other community initiative is almost entirely dependent upon the participation of its members. What you get out of this magazine is what you as a community put into it. So please get involved, submit your best articles, and do not hesitate to contact us with your suggestions.

Scott E. Delman, GROUP PUBLISHER



Group Term Life Insurance**

10- or 20-Year Group Term
Life Insurance*

Group Disability Income Insurance*

Group Accidental Death &
Dismemberment Insurance*

Group Catastrophic Major
Medical Insurance*

Group Dental Plan*

Long-Term Care Plan

Major Medical Insurance

Short-Term Medical Plan***

Who has time to think about insurance?

Today, it's likely you're busier than ever. So, the last thing you probably have on your mind is whether or not you are properly insured.

But in about the same time it takes to enjoy a cup of coffee, you can learn more about your ACM-sponsored group insurance program — a special member benefit that can help provide you financial security at economical group rates.

Take just a few minutes today to make sure you're properly insured.

Call Marsh Affinity Group Services at 1-800-503-9230 or visit www.personal-plans.com/acm.

3132851 35648 (7/07) © Seabury & Smith, Inc. 2007

The plans are subject to the terms, conditions, exclusions and limitations of the group policy. For costs and complete details of coverage, contact the plan administrator. Coverage may vary and may not be available in all states.

*Underwritten by The United States Life Insurance Company in the City of New York, a member company of American International Group, Inc.

**Underwritten by American General Assurance Company, a member company of American International Group, Inc.

***Coverage is available through Assurant Health and underwritten by Time Insurance Company.

AG5217

MARSH

Affinity Group Services
a service of Seabury & Smith

Knuth's Art of Recovering from Errors

I REALLY ENJOYED Donald E. Knuth reminiscing in Edward Feigenbaum's interview with him ("The 'Art' of Being Donald Knuth," July 2008). Who else would have moved to Stanford University to slow down?

Knuth's self-effacing modesty notwithstanding, I would like to challenge anyone to examine the literature preceding Knuth's contribution to algorithms and compiler theory. There was good work, but it was Knuth who set the field on its feet. I'm now looking forward to learning how he "solves the problem of typesetting" (see Aug. 2008). I personally made the mistake of using Microsoft Word for my Ph.D. thesis (completed 1996). More recently, I decided that the topic—efficient object-oriented programming for shared-memory multiprocessors—was of interest again due to the rise of multicore computers and reformatted it as a book. What a nightmare. I promised myself I'd use LaTeX for every Ph.D. I ever write again.

Philip Machanick, Taringa, Australia

Dependable Design and the Consequences of Failure

Leah Hoffman's news article "In Search of Dependable Design" (July 2008) was a good overview of some of the issues affecting software and system reliability, explaining how system dependability might be improved through good engineering practice. This is similar to a subject I covered in *Technology Review* (Apr. 1987) on software reliability.

However, Hoffman did not adequately discuss the theoretical limits that add to the risk of real-time, interactive applications. Peter Wegner's *Communications* article "Why Interaction Is More Powerful Than Algorithms" (May 1997) pointed out that interaction systems are not only difficult to verify but also formally incomplete, impossible to verify. Where the risk of system failure cannot be further reduced due to such limits, effort must be directed in-

stead at reducing the consequences of failure.

Ron Enfield, Cherry Hill, NJ

Technology Supports, Doesn't Supplant the Social Compact

In "Information Accountability" (June 2008), Daniel J. Weitzner et al. so exaggerated the good points they made that they created a false dichotomy between computing technologies (such as encryption) and societal conventions and laws, summing up in the article's final sentence: "Technology will better support freedom by relying on these social compacts than by seeking to supplant them." They were wrong. Technology does not seek to supplant social compacts but indeed seeks to add to them, legitimately.

Alex Simonelis, Montreal

Proud to Be a Member

My usual routine on finding *Communications* in my mailbox is to spend a few minutes flipping through the pages, then throw it on top of a pile of other unread magazines ready for recycling. However, the new cover typography (July 2008) immediately caught my eye. The title font was crisp and simple, recalling the all-caps, squarish characters of a teletype from the 1950s, very post-modern.

Flipping through the pages, I was struck first by the new design—clean and cool yet serious, information-dense, with narrow margins. And with articles on model checking, XML, transactional memory, and more, I actually wanted to read it. Beginning with the interview with Emerson, Clarke, and Sifakis, I then read the article on the history of IT in India and the article on quantum computing before I came to the Editor's Letter describing the makeover.

I had somehow missed that *Communications* was being overhauled. All I can say is: The editors, staff, and contributors have done an amazing

job. I've never really spent time thinking about ACM as an organization, but having a professional journal of this quality makes me proud to be a member.

Henry Kautz, Rochester, NY

Congratulations on the first issue of the new *Communications* (July 2008). The lineup of all-star authors (some of my favorite people in the field) and the relevant topics were amazing. It's wonderful to see computing's flagship journal return to such a high level of quality. The blend of code, programming-model, language, architecture, education, legal, and business topics represented an inspirational cross-section of the industry.

The issue reminded me how much fun it is to be a computer scientist. Reading it from cover to cover made me even more proud to sign my email...chaiken@acm.org.

David Chaiken, Menlo Park, CA

Correction

Due to a copy-editing error, a comma was added to the name of one of the authors of the Research Highlights paper "Wake Up and Smell the Coffee: Evaluation Methodology for the 21st Century" (Aug. 2008). Our apologies to J. Eliot B. Moss.

Communications welcomes your opinion. To submit a Letter to the Editor, please limit your comments to 500 words or less and send to letters@cacm.acm.org.

Current Features

We're All Ears

Our readers are a most opinionated audience and we value those opinions. Feel free to share your thoughts or get something off your chest. There are many ways to present your insights, comments, complaints, or questions to *Communications*:

- ▶ Read something that's got you thinking? Set you reeling?
Send a letter to the editor: letters@cacm.acm.org.
- ▶ Questions about your monthly digital edition?
Send them to: cacmfeedback@hq.acm.org.
- ▶ Email the Publisher and Editor-in-Chief directly;
their addresses are atop the *Communications* masthead on page 4.
- ▶ To contact other staff members, click on the "CACM Staff" link on the home page—<http://cacm.acm.org/>—or go directly to <http://cacm.acm.org/communications?pageIndex=6/>.

Still Everywhere

ACM's popular online publication, *Ubiquity*, remains as thought-provoking as ever under its new leadership. The preeminent Peter J. Denning has taken over as *Ubiquity*'s editor, replacing John Gehl who served in that position for almost a decade. See what Denning, a longtime *Communications* columnist, former ACM president, and current chair of the Computer Science Department at the Naval Postgraduate School in Monterey, CA, and his newly assembled editorial board have in store at www.acm.org/ubiquity.



Mobile Magazine

The Digital Edition of *Communications* is viewable on your iPhone or iPod Touch. So, too, are other ACM publications, including *Queue*, for forward-looking practitioners; *interactions*, for HCI (human-computer interaction) fans; *netWorker*, for networking technologists; and *Crossroads*, ACM's student publication. These and 84 other titles are available in Texterity Inc.'s electronic format that re-creates each page of a printed magazine. Find them at iphone.texterity.com.

ACM Member News

FROM COMPUTER GEEK TO COMPUTER CHIC

ACM and the WGBH Educational Foundation have won a two-year grant from the National Science Foundation to improve the image of computer science among high school students. Launched in June, the New Image for Computing project will create a national outreach and communications plan to create interest among high school students, particularly African-American boys and Latina girls, about pursuing computer science as a college major and a subsequent career choice. "Obviously there's a whole pool of talent that is not aware of what computing can do to help them achieve their goals in healthcare or whatever career they choose to pursue," says Jill Ross, director of the Image of Computing Task Force, which is leading a national coordination effort to provide a realistic view of opportunities in computing.

GÖDEL PRIZE WINNERS

Daniel A. Spielman and Shang-Hua Teng were awarded the 2008 Gödel Prize by ACM's Special Interest Group on Algorithms and Computing Theory and the European Association for Theoretical Computer Science for developing a rigorous framework to explain the practical success of algorithms on real data and real computers that could not be clearly understood through traditional techniques.

OOPSLA '08 CONFERENCE

If you're interested in the future of objects in software development, be sure to attend OOPSLA '08: ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, which will be held in Nashville, TN, from Oct. 19–23. The conference will address current software challenges such as programmer productivity, security and reliability, ultra large-scale systems, and evolving hardware platforms. For more information, visit <http://www.oopsla.org/oopsla2008>.

Finding Diamonds in the Rough

Spectral graph theory has proven to be very useful for text search and retrieval and for refining predictive-analysis systems.

THERE IS A little-known approach to information analysis that has built the foundation for many of the information technologies that we now consider to be givens of the 21st century. The strategy, called spectral graph theory, is well known among mathematicians and those working with massive data sets, but has not received a great deal of credit in the mainstream media as being an important method for understanding key relationships in data sets consisting of millions or even billions of nodes. With roots in the early 20th century, spectral graph theory and the corresponding interpretative method of spectral analysis were initially used as a theoretical approach to solving specialized math problems in which relationships between certain classes would otherwise be difficult to ascertain.

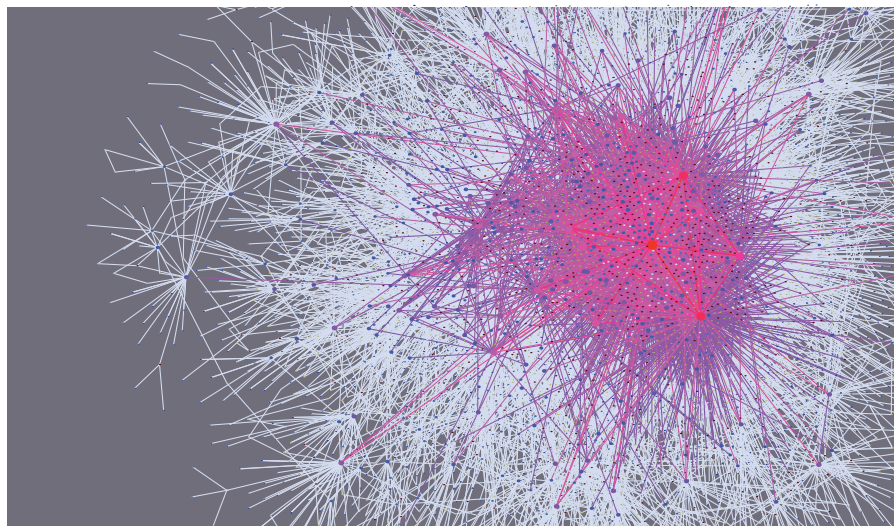
The origin of spectral graph theory can be traced to Markov chains, harmonic analysis, statistics, and spectral geometry, with mathematicians asking Zen-like questions such as “How can you hear the shape of a drum?” In the intervening years, spectral graph theory was applied in several areas, with IBM researcher Alan J. Hoffman even using

the technique in the 1970s to partition circuitry on silicon chips. In the 1980s, Microsoft’s Susan Dumais, the University of Colorado’s Tom Landauer, and their colleagues used spectral analysis to cluster documents in a technique called latent semantic analysis. However, it wasn’t until the 1990s that spectral graph theory became one of the most important tools in understanding how to conduct text search and retrieval more efficiently.

Spectral graph theory, which might be characterized as a kind of advanced

Boolean logic on steroids (in the sense that it can help simplify extremely complex relationships among class members that are also members of other classes), has become useful not only to those working in Web search, but also to those refining predictive-analysis systems of the type used to determine what books you might find most interesting or what movies you might like to watch.

In her early work with spectral analysis, Fan Chung, an Akamai professor in Internet mathematics at the University of California, San Diego, applied the theory to light patterns to help determine the molecular composition of stars. Chung is now taking spectral analysis in new directions for Web search, such as how to analyze and improve Google’s PageRank, which was introduced in 1998 and based on a math strategy called random walks. “One of the main applications of spectral graph



theory is to analyze the rate of convergence of random walks,” says Chung. “So you can see it takes just one further extension to analyze PageRank.”

Chung points out that the algorithms used in spectral graph theory have become so efficient that it is now possible for graduate students to apply the theory to data sets consisting of millions of nodes. “Five years ago I would have never imagined my students would be able to deal with millions of data points,” she says. “The power of the algorithm allows us to deal with this many data points on a single, dedicated PC.”

One of the metaphors that Chung uses to characterize spectral graph theory is a sandbox. The traditional way of identifying the relationships between grains of sand in a sandbox is to string them together, one by one. Spectral graph theory essentially makes such work much more efficient by instantly “cutting” to the relationships between all the grains of sand so it is possible to quickly identify, as Chung puts it, “diamonds in the rough.”

Identifying Data Patterns

To understand how spectral graph theory cuts to these relationships, you must understand eigenvalues and eigenvectors, which are values placed on a group of data points. Unlike Boolean operators, which deal with either-or classes that have no relative weights, eigenval-

ues and eigenvectors enable mathematicians and others working with spectral graph theory to apply finely tuned weights to members of a population with the goal of identifying interesting data patterns. In a social network, for example, an eigenvalue is a number indicating a particular pattern or cluster. The pattern could be men and women, for example. The larger the eigenvalue, the more important the pattern. An eigenvector would be a particular weight applied to the men-women partition, with plus values applied to women and minus values applied to men.

One strength of spectral graph theory is the ability to apply multiple eigenvalues and eigenvectors to the equation, so you might have a value for, say, conservatives and liberals or those who read magazines and those who don't. In the end, every member of the social network has a numeric score. The idea is to cut to the eigenvalues that are important, then apply eigenvectors to determine what the clusters are. Rather than working with either-or relationships, which make it relatively difficult to efficiently apply gradient values, spectral graph theory enables researchers to tease out useful clusters by applying multiple values to each member of a population.

While the most well-known application of the technology today is in Web search, Milena Mihail, an associate professor in the College of Comput-

ing at the Georgia Institute of Technology, points out that spectral graph theory can be applied to just about any technology outside the Web to reveal which clusters in a massive data set are the most important. “The power of the method does not come only from the fact that it has some mathematical justification,” she says. “It comes from the fact that, computationally, it is very easy to implement. Doing spectral analysis on a very large data set can be done in almost linear time, on the spot.”

According to Mihail, spectral graph theory has a disadvantage in that it needs numeric stability. Mihail says the next major breakthrough in spectral analysis might be for distributed applications, such as peer-to-peer networks, in which little numeric stability exists because no central authority has total knowledge of the network.

In talking about spectral graph theory and the early days of Web search, Mihail points to the work of Prabhakar Raghavan, head of research at Yahoo!, and Cornell University's Jon Kleinberg as being fundamental to the modern-day concept of automating search results through the application of spectral analysis. Raghavan headed up the Computer Science Principles department at IBM's Almaden Research Center and led IBM's Clever Project on Web search and page popularity. The Clever Project has received a great deal of attention for developing efficient

Eigenvectors applied to pages for the search query jaguar*.

(jaguar*) Authorities: principal Eigenvector

.370	http://www2.ecst.csuchico.edu/_jschlich/Jaguar/jaguar.html	
.347	http://www-und.ida.liu.se/_t94patsa/jserver.html	
.292	http://tangram.informatik.uni-kl.de:8001/_rgehml/jaguar.html	
.287	http://www.mcc.ac.uk/dlms/Consoles/jaguar.html	Jaguar page

(jaguar jaguars) Authorities: 2nd nonprincipal vector, positive end

.255	http://www.jaguarsnfl.com/	Official Jacksonville Jaguars NFL Web site
.137	http://www.nando.net/SportServer/football/nfl/jax.html	Jacksonville Jaguars home page
.133	http://www.ao.net/_brett/jaguar/index.html	Brett's Jaguar page
.110	http://www.usatoday.com/sports/football/sfn/sfn30.htm	Jacksonville Jaguars

(jaguar jaguars) Authorities: 3rd nonprincipal vector, positive end

.227	http://www.jaguarvehicles.com/	Jaguar Cars Global home page
.227	http://www.collection.co.uk/	The Jaguar collection official Web site
.211	http://www.moran.com/sterling/sterling.html	
.211	http://www.coys.co.uk/	

ways to determine search results through measuring link popularity.

Kleinberg, a professor of computer science at Cornell, also worked on IBM's Clever Project. Kleinberg's research in this area relies on what he calls hubs and authorities: The way to find good hub pages is to find good authority pages that they link to, and the way to find good authorities is to find good hubs that link to the authorities. In a sense, this circular problem amounts to a mathematical version of Oroboros, the mythological serpent that swallows its own tail. And it is here that spectral graph theory made one of its most important contributions to Web search by offering a way to break that circularity.

Kleinberg explains that overcoming this problem involves using an eigenvector to determine the relative value of a node's endorsement. If one node endorses its neighbor, the quality of the endorsement would be equivalent to the sum of the qualities of every node endorsing that neighbor node, and so on. You can keep playing this hub-and-authority game all the way out to infinity. Eventually, though, things start to stabilize. "What they start to stabilize on is an eigenvector of the graph," Kleinberg says. "In Web search, an eigenvector of the graph is essentially the infinite limit of a sequence of increasingly refined estimates of quality."

The table here, derived from Kleinberg's seminal 1997 IBM research report titled *Authoritative Sources in a Hyperlinked Context*, serves as a good example for how automated semantic separation can be accomplished with eigenvectors. On the most positive weight nodes of the first eigenvector for the search "jaguar," spectral analysis turned up sites related to the Atari Jaguar video game console. On the second eigenvector, spectral analysis turned up sites related to the Jacksonville Jaguars football team. And on the third eigenvector, spectral analysis turned up Jaguar car dealers. The numbers in the table represent coordinates from the first few eigenvectors; thus, the pages with large coordinates in each of these eigenvectors correspond to results for different meanings of the query term. This is the sense in which the different eigenvectors are serving to distinguish between different meanings of a query.

According to Kleinberg, the ability

Spectral graph theory is the natural mathematical language for talking about ranking on the Web, says Jon Kleinberg.

to break through the problem of circularity with eigenvectors is why spectral graph theory is the natural mathematical language for talking about ranking on the Web. "But I think there is more basic research that needs to be done because it's not clear that we've seen the full power of the technique," he says.

Kleinberg points out that most applications of spectral graph theory have been related to finding global, large-scale features, such as the highest-quality Web pages or the most significant user clusters in a customer database. He suggests that spectral analysis is not as naturally suited to finding needle-in-a-haystack details, such as 100 interesting Web pages in billions of Web pages or a few unusual purchases in terabytes of customer data. And when it comes to the theory itself, Kleinberg says that in some cases there are no guarantees for the quality of the solutions that spectral analysis offers. "So, trying to actually prove some guarantees on the performance of algorithms based on spectral analysis is a very nice open question," he says. "And in the process of trying to find proofs for the performance of spectral algorithms, we might in the process invent new algorithms."

Breakthroughs or not, it seems likely that spectral analysis will remain a favorite tool among mathematicians and computer scientists working with vast amounts of data, including those doing work in computer vision, image recognition, and any other area of research that might benefit from advanced pattern-recognition strategies. **Q**

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

Science

Natural Storage

DNA is seen by many researchers as the classic information storage system. Just four bases (adenine, thymine, guanine, and cytosine) are required to code ongoing combinations of multiple amino acids that ultimately steer the cellular machinery. Inspired by the notion that DNA could be used to store nonbiological data, a team of researchers in Japan has created the first DNA strand made from artificial bases. The researchers, in a paper recently published in the *Journal of the American Chemical Society*, explain all the components of their DNA product are nonnatural, yet they spontaneously form duplexes with the corresponding opposite base, and these bonds are very similar properties to those found in natural DNA. The team hopes this artificial DNA could offer a range of real-world applications, from using DNA to store data, to using it in biomedical and nanotechnology settings.

Computer Science

CS Award Winners

IEEE Honors

Gordon E. Moore will receive the IEEE Medal of Honor at the upcoming IEEE Annual Honors Ceremony, Sept. 20 in Quebec City, Canada. Other individuals to be recognized for work that "improved the world in which we live" are: Ralph H. Baer, Sir Timothy Berners-Lee, Joseph Bordogna, Don Coppersmith, Teuvo Kohonen, Leslie Lamport, Chrysostomos L. Nikias, Richard F. Rashid, Raj Reddy, and Alan Jay Smith.

EATCS Award

Leslie G. Valiant received the 2008 EATCS Award from the European Association for Theoretical Computer Science in recognition of his distinguished career in theoretical computer science.

Ubiquitous Video

Scalable and distributed video coding offers the promise of two-way, real-time video.

LARISSA, A BRAZILIAN foreign-language student studying in Tokyo, gets a call on her cell phone just as she arrives at her apartment after classes. She peers at the phone's display and sees her mother sitting in the living room of the family's home in São Paulo, plus a blinking blue dot indicating the call is a live, two-way video stream. Larissa flips open her phone.

"Mama, do you like my new haircut?" Larissa asks as she lets herself into her apartment. "Is it too short?"

"No, it looks terrific," says her mother. "I have some video of your father's birthday party. Please turn on your TV."

"Okay," replies Larissa, who points her cell phone at the 50-inch, flat-panel television on her living room wall and pushes a button. The television flashes awake, picks up the video stream from the phone, and displays a high-quality video of her family celebrating her father's 49th birthday at his favorite restaurant in São Paulo.

One phone call, one stream of information. The cell phone takes only the data it needs for its two-inch display while the 50-inch television monitor takes far more data for its greater resolution—all from the same video stream.

Welcome to the future world of scalable, distributed video.

Digital video coding compresses the original data into fewer bits while achieving a prescribed picture quality, which it accomplishes largely by eliminating redundancies. Image data for a static background object, for instance, is stored just once, with subsequent frames merely pointing back to the original and registering only incremental changes.

Today's video coding paradigm exploits temporal and spatial redundancies—think of them together as repetitive elements over time—with a series of predictions, a set of represen-



tations, and a slew of cosine calculations. The goals are to remove the details the human eye can't see (whether they're too fast, dark, or small), set aesthetic rules (such as color and aspect ratio), tailor the bit and frame rates for the highest picture quality at the lowest file size, and save as much bandwidth as possible.

A video stream is broken up into pictures that are not necessarily encoded in the order in which they are played back. Encoders append such commands as "for blocks 37–214, duplicate the same blocks in the last frame," and quantize the transform coefficients to control for the limitations of human visual perception. Finally, entropy coding acts to control the statistical redundancy of the resulting coded symbols.

It's not quite instant, but in fairly short order video encoders produce a digital video file, a fraction of its original size, for an iPod, laptop, or cell phone. And with advances in scalable and distributed video coding, two-way,

real-time video, such as Larissa's conversation with her mother, is becoming a reality.

Robust Encoding

Hybrid coding, which leverages both the temporal/predictive and frequency domains, is the basis for most current video standards. It does the hard work at the encoding step, resulting in complex encoders but just basic decoders.

A downlink model of a few encoders serving many distributed decoders serves applications for TV and cable broadcasting and on-demand Web video very well, but it makes decoder complexity its focus. Today's challenge, on the other hand, is the proliferation of wireless mobile devices—from cell phones and Internet tablets to laptops—that rely on up-links to deliver data. This requires capable device-based encoders.

In addition to robust encoding, these emerging applications require improved compression and increased

resistance to packet losses. New scalable and distributed coding solutions promise to deliver all of this—and much more.

A pair of Swiss-based standards organizations, International Organization for Standardization and International Telecommunication Union (ITU), formed a Joint Video Team in 2001 to develop a network-friendly video standard. Completed in 2003 and subsequently refined, H.264/AVC (Advanced Video Coding) attained measurably superior performance over existing standards. With the uplink model in ascendancy, there is continuing development in two promising areas: scalable video coding (SVC), an extension of the H.264/AVC standard, and distributed video coding (DVC).

An example of video scalability is when a “server has this 20Mbps coded video and you have a connection that can deliver 10Mbps,” says Gary Sullivan, a video architect with Microsoft and chair of the ITU-T Video Coding Experts Group. “If the video is encoded in a scalable way, the server can take just the subset of the data that represents the lower quality and give you that.”

Video data is delivered in packets, and if the video is not coded in a scalable manner, there’s basically very little a person can do other than decode all of them, notes Sullivan. However, if the video is encoded in a scalable way, then some packets belong to the base layer and some packets belong to the enhancement layer. Sullivan muses that it’s possible to create a bitstream with 10 layers, covering a wide range of decoders. “It’s a nice concept, but

“We know where [distributed video coding] may arrive from a theoretical point of view, but we still don’t know how to arrive there in practice,” says Fernando Pereira.

has been difficult to achieve,” he says.

Charting a New Course

A professor at the Electrical and Computers Engineering Department at Portugal’s Instituto Superior Técnico and the chair of many ad hoc video standards groups, Fernando Pereira is trying to chart video’s course from scalable to distributed. Not only will there be the multiple layers from SVC, but the new distributed video encoding will dynamically divvy up the work between encoders and decoders.

Pereira likens progress in the field of video coding to paleontologist Stephen Jay Gould’s description of “punctuated equilibrium” in evolution during which periods of stasis are interrupted by flurries of “creative destruction” and rapid change.

Video coding’s state of the art in

the early 1970s was represented by the Slepian-Wolf theorem that describes lossless coding—a way to reduce file sizes without losing any bits—with rather small compression factors. By 1976, Abraham Wyner and Jacob Ziv had derived the Wyner-Ziv theorem that essentially defines the conditions under which the picture quality can be achieved even when the coding process is not lossless.

Because it does not delete irrelevant information, the Slepian-Wolf theorem by itself has little practical application in video compression today. However, the Slepian-Wolf and Wyner-Ziv theorems together suggest the potential to compress two signals in a distributed way, with two separate encoders supplying a single joint decoder, says Pereira. He is confident this approach can achieve “a coding efficiency close to that of the predictive, joint encoding and decoding schemes” now in widespread use.

As opposed to conventional coding, in DVC the task of motion estimation is performed only on the decoder side to generate motion-compensated predictions for each input frame. The coding efficiency of a DVC scheme is judged to a great degree on the quality of these predictions.

The new DVC model promises substantial advantages for existing and emerging applications. They include flexible resources (DVC allocates varying amounts of encoder complexity to the decoder, which results in low encoder complexity and low battery consumption), improved resilience (DVC codecs do not rely on repetitive prediction loops, so channel in-

Workplace Technology

Keeping Focus Amid the Distractions

Today’s cubicle dweller switches gears every three minutes, moving from one task to *anything* else bombarding his or her attention. Indeed, the office environment has become such a breeding ground for interruptive technologies like emailing, cybersurfing, and IMing, that disruptions are now consuming as much as 28% of the average U.S. worker’s day and

sap productivity by as much as \$650 billion a year, according to Manhattan-based business research firm Basex.

In response, a growing number of tools and technologies are emerging to help keep workers on task. Microsoft is working on some remedies for attention disruption disorder that include AI systems that observe humans at work and

build models that predict the cost and benefit of interrupting someone, *BusinessWeek* reports. A prototype of an email triage program called Priorities ranks messages in order of perceived importance. The Outlook Mobile Manager enables Outlook to recognize urgent messages. And Bounded Deferral holds messages until a recipient is ready for a “cognitive break.”

IBM is also on the attention management track, now testing a prototype IM answering machine known as IMSavvy that can “sense” when a worker is too busy to answer calls or messages and will relay that sentiment to would-be interrupters. The system also offers a whisper option, flickering text on a worker’s screen even if the worker has instructed the system to withhold messages.

terference errors do not propagate over time), multiview independence (when used in a multiview video context, DVC encoders do not jointly process multiple views and thus do not need inter-camera, inter-encoder communication, saving energy), and codec-independent scalability (in current scalable codecs, a prediction approach from lower to upper layers requires the encoder to know the coding solutions for each layer, and the DVC approach allows each layer to use a discrete codec, unknown to the encoder, as knowledge of every layer is no longer necessary).

These benefits will positively impact video-related applications such as mobile videoconferencing and video email. “The future will tell us in which application domain the dis-

tributed source coding principles will find success,” says Pereira.

Video Everywhere

Although Pereira sees important roles for academia and industry, “DVC is still very much an academic exercise with very few companies involved,” he says. “MPEG [the family of standards used for coding audiovisual information] is not involved at all because it is too early to think about any standardization, and we still don’t know what the best solution may be.”

“With the continuing convergence of Internet, cable-based technologies, and wireless, bandwidth should also increase and we’ll be seeing more on-demand and live video applications very soon,” says Kevin Bee, CEO of Uptime Video, a video encoding firm based in Thousand Oaks, CA. This growing convergence has already led Adobe to include H.264 compatibility in its Flash Player 9, a move that has exponentially extended the codec’s reach.

“We know where DVC may arrive from a theoretical point of view, but we still don’t know how to arrive there in practice,” says Pereira.

Sullivan concurs. “H.264 itself gets easier to implement over time, but it will take a lot of work to make a better compression-capable codec,” he says. “We’re not there yet, and won’t be for several years at least.”

One major area of scientific research is human cognition. “Audio people had to enter this area earlier and deeper because the amount of redundancy in audio is much lower than in video, and they had to deal with irrelevancy in a more efficient way,” says Pereira. Clearly, he concludes, a better understanding of visual perception and the manner in which the human visual system responds to compression are among the most important next steps.

“The bottom line is that it is time for research and hard work,” says Pereira. “We should not go too fast in terms of making products so as to avoid ‘killing the goose that laid the golden egg.’ But, honestly, I don’t even know if there’s a goose yet.” □

Logan Kugler is a Los Angeles-based freelance writer who writes about business and technology.

Information Technology

Goodbye, Computer Mouse

The days of the computer mouse are nearing their end, Gartner analyst Steve Prentice has told the BBC News. Prentice expects the functionality of the mouse to be gradually replaced during the next three to five years by emerging alternative user interfaces that rely on facial recognition, movement, and gestures.

Prentice says the mouse has staying power in the desktop computer environment, but believes that “for home entertainment or working on a notebook, it’s over.”

He notes that Apple, Intel, and Microsoft are now promoting gestural interfaces for future computer use and that NEC, Panasonic, and Sony are demonstrating applications that use facial and movement recognition.

“With the [Nintendo] Wii you point and shake and it vibrates back at you so you have a two-way relationship there,” says Prentice. “The new generation of smart phones like the iPhone all have tilting mechanisms or you can shake the device to do one or more things. Even the multi-touch interface is so much more powerful and flexible than in the past allowing you to zoom in, scroll quickly, or contract things.”

Of course, not everyone agrees with Prentice. “The death of the computer mouse is greatly exaggerated,” says Rory Dooley, senior vice president and general manager of Logitech’s control devices unit, who notes that much of the developing world has still to get online. “There are around one billion people online, but the world’s population is over five billion.... The mouse will be even more popular than it is today as a result.”

Invented by Douglas Engelbart of the Stanford Research Institute, the computer mouse will celebrate its 40th anniversary later this year. Engelbart, recipient of the 1997 ACM A.M. Turing Award, never received any royalties for his invention, in part due to its patent expiring in 1987 before the widespread popularity of personal computers.

Coming Next Month in COMMUNICATIONS

An editorial debate on e-voting

Natural computing

Topology of dark networks

Code spelunking

A conversation with Pat Selinger

GPU architectures

Scene completion using millions of photographs

Geometry, flows, and graph partitioning algorithms

The enrollment crisis in Canada

Q&A with Daphne Koller, winner of the first ACM-Infosys Foundation Award

And the latest news on E2E systems and cryptography, green computing, and the Deep Web

Privacy Matters

As concerns about protecting personal data increase, differential privacy offers a promising solution.

OPEN A NEWSPAPER or a Web browser and you're certain to encounter a spate of stories about the misuse or loss of data and how it puts personal information at risk. Over the last decade, as computers and databases have grown ever more sophisticated, privacy concerns have moved to center stage. Today, government agencies worry about keeping highly sensitive financial and health data private. Corporations fret over protecting customer records. And the public grows ever more wary—and distrustful—of organizations that handle sensitive data.

"Privacy issues aren't about to go away," observes Adam Smith, an assistant professor in the computer science and engineering department at the Pennsylvania State University. "One problem we face is that 'privacy' is an overloaded term. It means different things to different people and a lot of issues hinge on context. As a result, it is extremely difficult to create effective solutions and protections—and to gain the trust that is necessary for respondents to answer sensitive questions honestly."

Some 220 million private records have been lost or stolen in the United States since January 2005, according to the Privacy Rights Clearinghouse, a San Diego, CA-based organization that tracks privacy issues. While no worldwide statistics exist, it's entirely apparent that a tangle of regulations, laws, and best practices cannot solve the problem. Worse, increasingly sophisticated tools make it possible to piece information together and glean details and facts about people in a way that wasn't imaginable a few years ago.

Now, a handful of researchers, mathematicians, and computer scientists are hoping to alter the landscape and frame the debate in new and important ways. Introducing a concept that has been dubbed "differential privacy," these data experts are seeking to use mathematical equations and algorithms to standard-



ize the way computers—and organizations—protect personal data while revealing overall statistical trends. The goal, says Cynthia Dwork, a principal researcher at Microsoft, is to ensure that an adversary cannot compromise data when he or she combines the released statistics with other external sources of information. "It's an extremely attractive approach," she says.

Connecting the Dots

The ability to collect and analyze vast data sets offers substantial promise. Sifting through medical data, genotype and phenotype connections, epidemiological statistics, and their correlation with events such as chemical spills or dietary and exercise patterns can help dictate public policy and find preventive strategies and cures for real people with real afflictions.

Yet, protecting privacy is an increasingly tricky proposition and one that confounds a growing number of organizations. Beyond the widely publicized hacker attacks and security lapses, there's an escalating threat of a person or organization assembling enough pieces of seemingly benign data—sometimes from different sources—to create a useful snapshot of a person or group. Kobbi

Nissim, an assistant professor of computer science at Ben-Gurion University, describes this approach as "connecting the dots." Oftentimes, it involves culling seemingly unrelated data from diverse and disparate sources.

It's not an abstract concept. When online movie rental firm Netflix decided to improve its recommendation system in 2007, executives emphasized that they would provide complete customer anonymity to participants. Netflix designed a system that retained the date of each movie rating along with the title and year of its release. And it assigned randomized numbers in place of customer IDs.

This seemed like a perfect system until a pair of researchers—graduate student Arvind Narayanan and professor Vitaly Shmatikov, both from the department of computer sciences at the University of Texas at Austin—proved that it was possible to identify individuals among a half-million participants by using public reviews published in the Internet Movie Database (IMDb) to identify movie ratings within Netflix's data. In fact, eight ratings along with dates were enough to provide 99% accuracy, according to the researchers.

This type of privacy violation—known as a linkage attack (attackers use innocuous data in one data set to identify a record in a second data set with both innocuous and sensitive data)—has serious repercussions, Dwork says. It could identify someone who is gay or has an interest in extremely violent or pornographic films. Such information might potentially interfere with a person's employment or affect his or her ability to rent an apartment or belong to a religious organization. "It could result in public humiliation," says Dwork, who notes that "the conclusion may be wrong. Partners share accounts. People buy gifts, and they may have some other reason for renting or buying certain movies."

It's not the first time such an event has taken place. In 2006, researchers

sifted through anonymized data of 20 million searches performed by 658,000 America Online subscribers. The researchers were able to cull sensitive information—including Social Security numbers, credit card numbers, addresses, and personal habits—by looking at all the searches of a single user (each user received a single randomized number). These same identification methods can be used for social networking sites and to parse through data contained in search engines, Dwork says.

The repercussions are enormous. For example, in the 1990s, a health insurance company that provided coverage for all state employees in the Commonwealth of Massachusetts released general data about the medical histories of anonymized individuals for general research purposes. Only the date of birth, gender, and ZIP code of residence was left in the data. However, a researcher, Latanya Sweeney, now an associate professor of computer science at Carnegie Mellon University, identified the medical history for William Weld, then the governor of Massachusetts. This was possible because the database contained only six people who had his same date of birth, only three of them were men, and Weld was the only person in his five-digit ZIP code.

What's remarkable, Dwork says, is that each data element alone isn't a privacy risk. "Most people would probably say, 'No big deal.' Yet, putting these three elements together is enough to identify approximately two-thirds of the population," says Dwork.

Preserving Privacy

As government agencies, research institutes, companies, and nonprofit orga-

Differential privacy appears to be the only approach that offers a solid and well-defined method for achieving privacy—without making any assumptions about the adversaries' strategy.

nizations search for ways to boost the value of their data, the pressure to develop better privacy-protecting methods and systems is increasing. "Despite good intentions and software tools designed to thwart breaches, breakdowns continue to take place," says Frank McSherry, a researcher at Microsoft Research Silicon Valley.

Privacy-preserving efforts have undergone a steady evolution during the last quarter-century. Statistics, security, cryptography, and databases have all emerged as topics of interest. However, actual solutions have remained elusive, largely because there's no way to guarantee data privacy with ad-hoc tools and methods. Cryptography, for example, is fine for protecting data from a security standpoint, but it does nothing to mitigate data mining and sophisticated analysis of publicly released or anonymized data. In fact,

mathematically rigorous methods have demonstrated that the 25-year-old concept of "semantic security" cannot be achieved for statistical databases.

Differential privacy, which first emerged in 2006 (though its roots go back to 2001), could provide the tipping point for real change. By introducing random noise and ensuring that a database behaves the same—independent of whether any individual or small group is included or excluded from the data set, thus making it impossible to tell which data set was used—it's possible to prevent personal data from being compromised or misused. Pennsylvania State University's Smith says that differential privacy can be applied to numerous environments and settings. "It creates a guideline for defining whether something is acceptable or not," he says.

Government, academic, and business leaders have shown some interest in differential privacy, although the concept is still in the early stages of development and implementation. Currently, differential privacy appears to be the only approach that offers a solid and well-defined method for achieving privacy—without making any assumptions about the adversaries' strategy. "There has been a lot of positive feedback about the concept, though it is clearly on the upward slope," McSherry says. "We believe that with further analysis, testing, and tweaking, differential privacy could emerge within the next several years as the gold standard for privacy." □

Samuel Greengard is a freelance writer based in West Linn, OR.

Computer Science

In Memoriam: Randy Pausch

To most of his followers around the world, Randy Pausch was known as the terminally ill college professor who taught millions how to live. His YouTube-based "Last Lecture," given a year ago at Carnegie Mellon University after doctors told him his pancreatic cancer had progressed and he had only months to live, became

an international sensation. His subsequent best-selling book of the same title was translated into over two dozen languages.

To the computing community, Pausch, who died on July 25 at the age of 47, will be forever known as a pioneer in virtual reality. His leadership in the Alice project revolutionized how introductory

programming can be taught. As co-founder of the Entertainment Technology Center at CMU, he developed a facility for bridging computer science with entertainment technology. Pausch, an ACM Fellow, was an active leader in SIGGRAPH and SIGCHI and a frequent contributor to many ACM publications; indeed, while

undergoing cancer treatments he co-authored two articles for *Communications'* special section on a science of games (July 2007).

On p. 19 we present a Q&A with Pausch conducted just weeks prior to his death. Like everything he did in life, he gave graciously of his time when it came to discussing computer science and his regard for students.

Wisdom from Randy Pausch

Randy Pausch, author of the best-selling *The Last Lecture* and a virtual-world innovator, on computer science, Alice, and teaching.

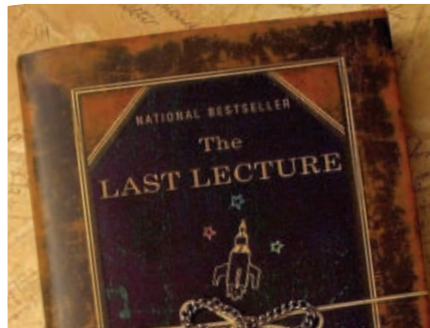
RANDY PAUSCH WAS known around the world for the inspiring “Last Lecture” he delivered last year—and for the battle he waged against terminal pancreatic cancer. To many people, Pausch, a professor of computer science at Carnegie Mellon University, served as a role model for working hard, overcoming obstacles, and achieving childhood dreams. Renowned as a passionate and creative teacher, Pausch won both the Karl V. Karlstrom Outstanding Educator Award from ACM and the ACM Special Interest Group on Computer Science Education Award for Outstanding Contributions to Computer Science Education in 2007.

Pausch was well known for his role as the driving force behind the Alice project, an innovative 3D computing environment that teaches students to program through an intuitive graphical interface. By addressing the challenge that syntax poses to novice programmers, Pausch opened the discipline to countless middle school, high school, and college students. Pausch’s own students remember his ability to help computer science come alive in person, making classes on subjects such as data structures compelling through well-chosen, motivating examples.

“Randy managed to collect this amazing group of people—some of the best people I’ve ever worked with,” says former student Caitlin Kelleher.

Last June, we asked Pausch to talk about what he hoped would be the legacy of his pioneering work in virtual reality and to share words of advice to students pondering a career in computer science. When the news came of Pausch’s death on July 25, the editors of *Communications* felt there could be no greater tribute than to share his own words about the joy he found in computer science with a like-minded audience and his hopes for the future of the field he cherished.

You’ve spoken of the importance of



never quitting—of continually pushing against brick walls and other obstacles. What additional advice might you give to tomorrow’s CS students?

Remember how quickly our field changes. That’s why you want to focus on learning things that *don’t* change: how to work well with other people, how to carefully assess a client’s real—as opposed to perceived—needs, and things like that.

What about advice for CS teachers and professors?

That it’s time for us to start being more honest with ourselves about what our field is and how we should approach teaching it. Personally, I think that if we had named the field “Information Engineering” as opposed to “Computer Science,” we would have had a better culture for the discipline. For example, CS departments are notorious for not instilling concepts like testing and validation the way many other engineering disciplines do.

Is there anything you wish someone had told you before you began your own studies?

Just that being technically strong is only one aspect of an education.

Let’s talk about Alice, the 3D programming environment you helped develop to teach kids how to program. What’s the most surprising thing you’ve learned from your work on the Alice project?

That no matter how good a teaching tool may be, it requires textbooks, lecture notes, and other supportive pedagogic materials before it can really become widely adopted.


Alice has proven phenomenally successful at teaching young women, in particular, to program. What else should we be doing to get more women engaged in computer science?

Well, it’s important to note that Alice works for both women *and* men. I think female-specific “approaches” can be dangerous for lots of reasons, but approaches like Alice, which focus on activities like storytelling, work across gender, age, and cultural background. It’s something very fundamental to want to tell stories. And Caitlin Kelleher’s dissertation did a fantastic job of showing just how powerful that approach is.

In a course on building virtual worlds, you required your students to create content without violence or pornography. Can you tell us a little more about how you reached that decision, and what the outcome was?

I just wanted them to do something *new*, and shooting violence and pornography were already heavily associated with virtual reality. Although I was impressed how many 19-year-old boys are flush out of ideas when you take those two off the table!

Do you have any predictions for the future of virtual reality or human-computer interaction (HCI)?

I think virtual reality needs better base technology; the Wii was a hit because of its input device, for example. North Carolina researchers got the tracking technology “good enough” almost ten years ago, but we are still waiting for a good, high resolution, lightweight, head-mounted display, which I think is critical. As for HCI, it is my hope that it stops being seen as a field “just for specialists,” and becomes something—like data structures—where every CS student should have at least one semester of HCI, just to understand the basic concepts. 

Based in Brooklyn, NY, Leah Hoffmann writes about science and technology.

ACM, Uniting the World's Computing Professionals, Researchers, Educators, and Students

Dear Colleague,

At a time when computing is at the center of the growing demand for technology jobs worldwide, ACM is continuing its work on initiatives to help computing professionals stay competitive in the global community. ACM delivers resources that advance computing as a science and profession.

As a member of ACM, you join nearly 90,000 other computing professionals and students worldwide to define the largest educational, scientific, and professional computing society. Whether you are pursuing scholarly research, building systems and applications, or managing computing projects, ACM offers opportunities to advance your interests.

MEMBER BENEFITS INCLUDE:

- A subscription to the completely redefined **Communications of the ACM**, ACM's flagship monthly magazine
- The option to subscribe to the full **ACM Digital Library**, with improved search functionalities and **Author Profile Pages** for almost every author in computing
- The **Guide to Computing Literature**, with over one million bibliographic citations
- Access to ACM's **Career & Job Center** offering a host of exclusive career-enhancing benefits
- **Free e-mentoring services** provided by MentorNet®
- **Full and unlimited access to over 3,000 online courses** from SkillSoft
- **Full and unlimited access to 1,100 online books**, featuring 500 from Books24x7®, and 600 from Safari® Books Online, including leading publishers such as O'Reilly (Professional Members only)
- The option to connect with the **best thinkers in computing** by joining **34 Special Interest Groups** or **hundreds of local chapters**
- **ACM's 40+ journals and magazines** at special member-only rates
- **TechNews**, ACM's tri-weekly email digest delivering stories on the latest IT news
- **CareerNews**, ACM's bi-monthly email digest providing career-related topics
- **MemberNet**, ACM's e-newsletter, covering ACM people and activities
- **Email forwarding service & filtering service**, providing members with a free acm.org email address and high-quality **Postini spam filtering**
- And much, much more!

ACM's worldwide network ranges from students to seasoned professionals and includes many of the leaders in the field. ACM members get access to this network, and enjoy the advantages that come from sharing in their collective expertise, all of which serves to keep our members at the forefront of the technology world.

I invite you to share the value of ACM membership with your colleagues and peers who are not yet members, and I hope you will encourage them to join and become a part of our global community.

Thank you for your membership in ACM.

Sincerely,



John R. White
Executive Director and Chief Executive Officer
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: ACACM29

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/serving/ethics.html>

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*. 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 _____



DOI:10.1145/1378727.1378736

Michael Cusumano

Technology Strategy and Management

The Puzzle of Apple

Given Apple's unique characteristics, should it strive to be a platform or a product leader?

ONE OF THE greatest product development companies in history is Apple, Inc., founded by Steve Jobs and Steve Wozniak in 1976.

Apple's list of "truly great" products—Jobs' promotional mantra for the Macintosh personal computer—is truly impressive, but the company has too often failed or chosen not to develop industrywide platforms. I will explain.

The Mac, introduced in 1984, pioneered the graphical user interface (albeit copied from Xerox) for the mass market. Other great Apple products include the first mass-market PC, the Apple II, introduced in 1977; the PowerBook, which in 1991 set the design standard for laptops; the unsuccessful though still-pioneering Newton PDA, first sold in 1993; and the iMac all-in-one "designer" PC, released in 1998. More recently, we have seen the iPod digital media player (2001), the iTunes music and other digital media service (2003), and the trendsetting iPhone (2007). Jobs does not take personal credit for all these products. He was absent from the company during

1985–1997 and returned only when Apple acquired his other company, NeXT Computer. That firm provided the basis for another hit Apple product released in 2001, the Mac OS X operating system. But Jobs created the design culture and hired or supervised the people (such as Jonathan Ive, chief designer of the iMac, the iPod, and the iPhone) most responsible for the company's current success and historical legacy.

But I have often wondered what the world would have been like if Steve Jobs had thought a bit more like his archrival, Bill Gates. Microsoft, founded in 1975, does not generally try to develop "truly great" products, although occasionally some are very good. Mostly, Microsoft tries to produce "good enough" products that can also serve as industry platforms and help bring cheap and powerful computing to the masses (and mega-profits to Microsoft). MS-DOS, Windows, and Office have done this since 1981.^a

^a See Michael A. Cusumano and Richard W. Selby, *Microsoft Secrets*, Free Press/Simon &

We can define the term "platform" as a foundation product or key technology in a system like the PC or a Web-enabled cell phone. A platform should have relatively open technical interfaces and easy licensing terms in order to encourage other firms to contribute complementary products and services. These external innovations create an ecosystem around the platform. The critical distinguishing feature of a platform is "network externalities": the more external firms in the network that create complementary innovations, the more valuable the platform becomes. This dynamic should cause more users to adopt the platform, more complementors to enter the ecosystem, more users to adopt, almost ad infinitum.^b (I say "almost" because there is some evi-

Schuster, NY, 1995.

^b For more discussion on platform dynamics, see Annabelle Gawer and Michael A. Cusumano, *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*, Harvard Business School Press, Boston, MA, 2002 and our recent article "How Companies Become Platform Leaders," *MIT Sloan Management Review* 49, 2 (Winter 2008), 28–35.



out in 1976 with its own product, the VHS recorder. Some observers thought it was technically inferior. But JVC and Matsushita treated VHS more as an industry platform. They incorporated feature suggestions from other firms, broadly licensed the new technology, provided essential components to licensees, and aggressively cultivated a complementary market in prerecorded tapes. The much larger number of firms licensing VHS encouraged tape producers and vendors to make and sell many more VHS than Betamax tapes. Users responded and bought more VHS machines, encouraging more firms to license the standard and then more tape producers and distributors and consumers to adopt VHS. Betamax disappeared.

We can tell almost the same platform vs. product story with the Macintosh. Apple chose to optimize the hardware-software system and monopolize revenues from the product. By contrast, a platform strategy would have meant licensing the Macintosh operating system widely and working openly with other companies and complement producers to evolve the platform and create applications for the mass market. Apple did not do this and remained (for the most part) the only producer of the Mac, keeping prices high (about twice the cost of an IBM-compatible PC using technology from Microsoft and Intel) and diffusion low. The Macintosh survived as a second standard with a few percent of the market only because it found two niches—desktop publishing as well as consumers who were willing to pay for an easier-to-use and more elegant product. But software applications producers—the major complementors of computer platforms—overwhelmingly chose to support the more broadly selling IBM-compatible machines.

Which brings me to more recent “truly great” products from Apple that have done better in the market. The iPod, with its unique “click wheel” interface and new touchscreen, is the best-selling music player in history, currently with about a 70% market share. It also has attracted complementary hardware that have made it more valuable, such as connectors for car or home-stereo systems, or add-ons that turn the iPod into an FM radio, digital recorder, or cam-

dence that too many complementors can reduce the incentives of new complementors to invest. ^c) Some markets with strong network externalities (such as through incompatible formats) and little opportunity for differentiation or niche strategies tend to evolve into “winner take all” or “winner take most” businesses, like Windows and Office for PC software, eBay for online buying and selling, or Akamai for Internet content hosting services. Google is moving in this direction as well for Internet search and contextual advertising. ^d

We have seen many platform battlegrounds in the history of technology, with prominent examples coming from the commercialization of electricity, radio, and television. My first platform-related research was the battle between

Sony—another great product company in its heyday—and Japan Victor Corporation over the home videocassette recorder (VCR).^e During 1969–1971, Sony engineers had compromised their technology goals to produce an earlier standard using $\frac{3}{4}$ -inch-wide tape, the U-Matic, in order to get the support of other firms. This product never succeeded with consumers because of its bulk and cost. When Sony engineers produced a smaller $\frac{1}{2}$ -inch tape version in 1975, the Betamax, Sony management tried to persuade other firms to adopt this product as the new standard. Sony refused to revise the design to accommodate other firms such as GE in the U.S., which wanted a longer recording time.

Japan Victor, backed by its giant parent Matsushita Electronics, came

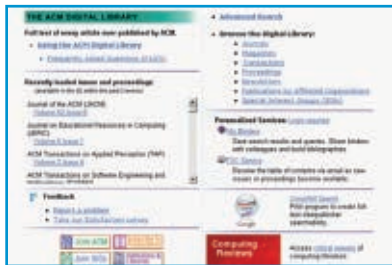
^c See Kevin Boudreau, “Too Many Complementors? Evidence on Software Firms,” Working Paper, HEC-Paris School of Management, November 2006.

^d For characteristics of “winner take all” markets, see Thomas Eisenmann, Geoffrey Parker, and Marshall W. Van Alstyne, “Strategies for Two-Sided Markets,” *Harvard Business Review*, October 2006, 1–10.

^e See Richard S. Rosenbloom and Michael A. Cusumano, “Technological Pioneering and Competitive Advantage: The Birth of the VCR Industry,” *California Management Review* 29, 4 (Summer 1987); and Michael A. Cusumano, Yiorgos Mylonadis, and Richard S. Rosenbloom, “Strategic Maneuvering and Mass-Market Dynamics: The Triumph of VHS Over Beta,” *Business History Review* (Spring 1992).

ACM Digital Library

www.acm.org/dl



The Ultimate Online INFORMATION TECHNOLOGY Resource!

- **NEW! Author Profile Pages**
- **Improved Search Capabilities**
- **Over 40 ACM publications, plus conference proceedings**
- **50+ years of archives**
- **Advanced searching capabilities**
- **Over 2 million pages of downloadable text**

Plus over one million
bibliographic citations are
available in the ACM Guide
to Computing Literature

To join ACM and/or subscribe to the Digital Library, contact ACM:

Phone: 1.800.342.6626 (U.S. & Canada)
+1.212.626.0500 (Global)

Fax: +1.212.944.1318

Hours: 8:30 a.m.–4:30 p.m., EST

Email: acmhelp@acm.org

Join URL: www.acm.org/joinacm

Mail: ACM Member Services

General Post Office

PO Box 30777

New York, NY 10087-0777 USA

era. Initially, however, Apple introduced the iPod as a closed system that worked only with the Macintosh and iTunes music warehouse and did not support non-Apple music formats or software applications. It was as if Microsoft produced Windows and then built Office but did not allow other software companies to build Windows-compatible applications. Eventually, under pressure, Apple opened up the iPod software (but not the hardware) to play some other music formats, but not those from Microsoft or Real. Apple also uses proprietary digital rights management (DRM) technology on the iPod and the iTunes store, creating problems with potential ecosystem partners as well as customers. (To its credit, though, Apple did introduce an iPod in 2002 compatible with Windows and then a Windows version of iTunes in 2003.)

Then we have the iPhone—probably the most exciting electronics product to hit the market since the Macintosh. This “smartphone”—a cell phone with many of the capabilities of a digital media player as well as a Web-enabled handheld computer—also boasts a remarkable user interface driven mainly by touch and virtual keyboard technology. But the original iPhone would not run applications not built by Apple, and it would not operate on cell phone networks not approved by Apple (initially only AT&T in the U.S., but later Deutsche Telekom/T-Mobile in Germany, Telefonica/O2 in the U.K., and Orange in France). Fortunately for consumers, hackers around the world found ways to unlock the phone and add applications. A black market also developed for “hacked” devices. This market pressure persuaded Apple that its great new product was becoming a platform and needed to be more open to outside applications. In March 2008, Jobs also announced that Apple would license Microsoft’s email technology to enable the iPhone to connect to corporate email servers. But Apple has yet to allow consumers to use the iPhone on any service network they choose.

If Steve Jobs and the rest of the Apple team had thought to make “great platforms” first and “great products” second, then it is possible that most PC, digital media, and smartphone users

Despite faster recent growth than Microsoft, Apple relies too much on the fleeting nature of “hit” products.

today would be using Apple products. Despite faster recent growth than Microsoft, Apple relies too much on the fleeting nature of “hit” products. Apple still is just half the size of Microsoft in revenues and much less profitable. Apple won the battle for digital media players but that product, like PDAs, is likely to disappear in favor of smartphones. Apple may yet win the smartphone battle but still trails RIM’s BlackBerry and Symbian/Nokia smartphones by a wide margin. We shall see how the market plays out, as Nokia, Samsung, and other firms introduce products that look and feel similar to the iPhone.

Which leads me to the puzzle alluded to in the title for this column: Is it possible for a company with Apple’s creativity, foresight, and independence to think “great platform” first and still produce “truly great” products? Based on Sony’s experience with VCRs, or Microsoft’s with MS-DOS and Windows, it is clear that platform companies must work with industry players and be willing to make technical and design compromises, as Nokia has done with the Symbian consortium. Jobs and other Apple managers have been acutely aware of the product versus platform challenge and have preferred not to follow an open platform strategy. But customers have eventually pressured Apple to open its products and it has done so without losing too much distinctiveness. This evolution suggests that Jobs and Apple could have pursued product and platform leadership simultaneously. Just a thought about what might have been. □

Michael Cusumano (cusumano@mit.edu) is a professor at the MIT Sloan School of Management and author of *The Business of Software*, Free Press/Simon and Schuster, 2004.



DOI:10.1145/1378727.1378737

George Neville-Neil

Kode Vicious Pride and Prejudice (The *Vasa*)

Navigating the well-traveled course of communication failure that often leads to engineering disasters.

Dear KV,

I teach computer science to undergraduate students at a school in California and one of my friends in the English department, of all places, made an interesting comment to me the other day. He wanted to know if my students had ever read *Frankenstein* and if not if I felt it would make them better engineers. I asked him why he thought I should assign this book and he said he felt that a book could change the way in which people think about their relationship to the world, and in particular to technology. He wasn't being condescending, he was dead serious. Given the number of Frankenstein-like projects that seem to get built with information technology, perhaps it's not a bad idea to teach these lessons to computer science undergraduates, to give them some notion that they have a social responsibility?

CS Prof

Dear CS,

While I have to agree in general with the idea that telling and retelling stories is a good way to teach people, I have to say that the idea of using Mary Shelly's novel for this is very much antiquated and unlikely to be effective in a computer science class. I, myself, was once forced through a "Computers and Society" course in college, and although we didn't read *Frankenstein* we were beaten over the head with a litany of how bad computers and technology were for society from a professor who was trivial to manipulate. All I had to do was agree

with her every utterance and write technology-bashing essays for her class to get an A. Was this an effective use of time?

Of course not, it was a show. If you really want to reach an audience you have to engage them with stories that you understand and can relate to their experience. When I think of the kind of story I want to tell to undergraduate students, I think of the *Vasa*, a ship and story that I think should be better known among engineers.

I first learned of the *Vasa* from a t-shirt at a conference in 1990. A company that a friend had started used the cross section of the ship to lampoon the ISO/OSI effort on network protocols. "Another 7 Layer Model That Failed" read the caption. The connection was that ISO had seven layers and the *Vasa* had seven decks, but when I found out why the *Vasa* had tragically failed I be-

came fascinated, because it was such a classic engineering failure story.

The *Vasa* was built between 1626 and 1628 for King Gustavus Adolphus of Sweden, who was, at that time, attempting to rule the Baltic Sea. In the 17th century, rulers were expected to be capable of more than just giving orders, so Adolphus not only organized wars, he also helped design the ships of his naval fleet. At the time Swedish warships had one deck of cannons on each side from which they fired fusillades at enemy ships, sometimes even hitting the other ships and damaging them. When the *Vasa* was commissioned, this single row of cannons was considered state of the art.

Some time during the construction of the ship Adolphus found out that the Poles had ships with two decks of guns, so he modified the design of the *Vasa* to have a second gun deck. This would



have made it the most powerful naval vessel of the time, capable of delivering a broadside of devastating proportions. The men he had contracted to build his ships attempted to explain that the ship had too little ballast to support two gun decks, and that the resulting ship likely would be unsafe to sail. The King insisted—just like, say, many project managers—that his orders should be followed. On a software project you can quit, but if the King is your boss you might lose more than your job—you might, say, lose your head—so the project went forward.

In 1628 the ship was finally ready for quality assurance (QA) testing. Seventeenth-century QA of ships was a bit different from what might happen today. Thirty sailors were picked and asked to run back and forth, port to starboard, across the deck of the ship. If the ship didn't tip over and sink, then the ship passed the test. You did not want to be on the QA team in 1628. After only three runs across the deck the *Vasa* began to tilt wildly and the test was canceled. The test may have been canceled, but not the project. This was the King's ship, after

all, and she would sail. And sail she did.

On August 10, 1628, in a light breeze, the *Vasa* set sail. She was less than a mile from dock when a stiff breeze knocked her sideways. She took on water, and sank in full view of a crowd of thousands of onlookers. Approximately 30 to 50 sailors were killed when they were either trapped in the ship or were unable to swim to shore.

In response to the catastrophe, the King wrote a letter insisting that incompetence had been the reason for the disaster. He was, of course, correct, but not in the way he might have envisioned. An inquest was held and the surviving members of the crew, the captain, and the ship builders were questioned as to the state of the crew and the ship at the time of the incident. The mostly unstated belief by the end of the inquest was that the design had been a failure and the designer had not listened to the builders about the shortcomings of the design. Of course, the King could not be held at fault, so the final verdict was an "act of God." As a related aside, the disaster was also a huge economic loss for Sweden.

Now, this story may not be as well written as *Frankenstein*, but it's a much more direct warning about engineering failures. I think the funniest or saddest part of this story is how modern it is. Nothing has changed since 1628. People still fail to communicate, leading to failures of disastrous proportions. Egos get in the way, mysterious supernatural forces are blamed for human failings. It's all kind of obvious in a really sad way.

In the 1960s the *Vasa* was raised from the bottom of the bay in which it had sunk and eventually placed in a museum in Stockholm. I visited the *Vasa* in 2000 as part of the SIGCOMM conference. The whole story is told there in the plaques on the walls. It's a museum all engineers ought to visit at least once.

KV

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.

ACM Journal on Emerging Technologies in Computing Systems



This quarterly publication provides comprehensive coverage of innovative work in the specification, design analysis, simulation, verification, testing, and evaluation of computing systems constructed out of emerging technologies and advanced semiconductors. Topics include, but are not limited to: Logic Primitive Design and Synthesis; System-Level Specification, Design and Synthesis; Software-Level Specification, Design and Synthesis; and Mixed-Technology Systems.

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



Cameron Wilson and Peter Harsha

DOI:10.1145/1378727.1378738

IT Policy

Science Policy Isn't Always About Science

What is the appropriate role and level of influence for science and technical advice in policy deliberations?

THIRTEEN YEARS AGO this month, a newly empowered Republican majority in the U.S. House of Representatives, committed to a “Contract with America” that included self-imposed austerity measures in an effort to reduce the size of government, took aim at a small legislative branch agency with the job of providing non-partisan, objective information to Congress about the impact of technology. By agreeing to shutter the Office of Technology Assessment (OTA) in 1995, Congress managed to save \$22 million in the roughly \$2 billion appropriations bill funding congressional salaries, support staff, and related operations—and committed what one critic has called a “stunning act of self-lobotomy.”^a

The OTA’s mission was to help resolve a challenging problem for policymakers—navigating the intersection of technical and scientific issues with policy-making. During its 23 years of operation it produced more than 750 reports on a wide variety of products from fusion energy to infertility. But Congress had begun to question the relevancy of the agency, noting that there were other entities like the General Accounting Office (now Government Accountability Office), the National Academies, and the Congressional Research Service that were providing ostensibly similar products to Congress.

a C. Mooney, “Requiem for an Office.” *Bulletin of the Atomic Scientists*, (Sept./Oct. 2005).

The Republican Chair of the House Science Committee complained that though the agency produced detailed and voluminous reports, they often lagged critical debates and languished on Congressional shelves.^b In addition, though the agency strived to produce reports that were scientifically rigorous and non-partisan, reports questioning the goals and technical feasibility of the Reagan Administration’s Strategic Defense Initiative anti-ballistic missile system soured the office to many of SDI’s supporters in Congress.

The Republican Speaker of the House, Newt Gingrich, railed against the office and its 150-member staff, claiming that what Congress needed was scientists and engineers speaking directly to members of Congress, not working through the filter of OTA’s “bureaucracy.” The OTA made for “bureaucratic science,” he later said.^c “Congress needs first-rate scientists talking to its members. It does not need congressional staff analysts talking to congressional staff members to develop staff-driven documents that are then presented to congressmen.”^d

Though OTA and its supporters an-

b W. O’Leary, “Congress’s Science Agency Prepares to Close its Doors.” *New York Times*, (Sept. 24, 1995), 26.

c Remarks before the Computer Science and Telecommunications Board, The National Academies, Oct. 6, 2005 (see www.cra.org/govaffairs/blog/archives/000419.html).

d See <http://freakonomics.blogs.nytimes.com/2008/03/14/newt-gingrich-answers-your-questions/>.

swered these criticisms—for example, many pointed out that while GAO, the National Academies, and CRS all produced reports, none were equipped or attempted to bridge policy and science in the way that OTA did; and the idea of members of Congress each having the wherewithal to identify appropriate voices from the technical fields on their own to glean from them the key points relevant to making policy was somewhat absurd—the politics of the moment overwhelmed them and the agency lost its funding.

The elimination of OTA set simmering a debate in policy circles about the role and prominence of science and technical advice in policy deliberations that continues today. Much of that discussion, including some that has appeared in previous issues of *Communications*,^e is focused on making a renewed case for a native technical assessment capability for Congress and refuting the claims of OTA’s critics. There have been several legislative attempts to revive the agency in the 13 years since its closing, but none have gained serious traction—though the FY 2008 Omnibus appropriation included \$2.5 million for a small technical assessment role for the GAO.

While it is easy to see this development as a net loss for the science community—and on balance, it likely is—

e J. Peha, “The Growing Debate Over Science and Technology Advice for Congress.” *Commun. ACM* 44, 12 (Dec. 2001), 29.

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

when thinking about its implications it is also easy to lose sight of a more fundamental truth about science and policy: the presence of a rational scientific argument does not guarantee that Congress will do the “right thing.”

In our experience as two people who serve at the interface between the science community and policymakers, we frequently hear the frustration from researchers and engineers about the failures of Congress to act in “rational” ways regarding new technologies. On everything from security, to privacy, to voting, to research funding, to intellectual property issues, there is a sense that “if they only had the right information, they’d make the right decision.” This is part of the impulse behind the move to restore OTA.

But what made OTA’s job so difficult, and what presents a broader challenge to the entire scientific community, is that the nexus of science and technology and public policy is layered over a political system. Decisions on public policy are not made in the abstract based on the best technical information; they are made by policymakers balancing numerous interests from numerous constituencies. Balancing these interests is a political calculus, not a scientific one. In developing it, scientific information is sometimes pivotal, sometimes sits on the sidelines, and sometimes ends up as a mix where compromise may or may not embrace a technical truth. It is difficult to predict what factors will motivate specific policy debates because the political system is ever changing. While there are always a good number of members of Congress who can see the importance of making the correct decision versus the political one, party politics demands a consistent view toward self-preservation. For better or worse, in the political calculus the paramount concern is how a decision impacts a member back home.

An apt example of the tension between the “right” course of action and politics in a current context is the fight for increased funding for the physical sciences (which, in Washington, D.C. parlance includes such non-life science fields such as computing, mathematics, chemistry, and engineering). In August 2007, Congress and the president overwhelmingly passed the bipartisan America COMPETES (Creating Opportunities to Meaningfully Promote

Excellence in Technology, Education, and Science) Act, which sought to bolster America’s innovation ecosystem by setting a goal of doubling over seven years of the research budgets at three key science agencies—the National Science Foundation (NSF), the Department of Energy’s Office of Science, and the National Institute of Standards and Technology (NIST)—and creating and expanding science and math education programs.

The COMPETES Act was modeled on the recommendations of a National Academies report chaired by former Lockheed Martin Chairman Norman Augustine called *Rising Above the Gathering Storm*. That report was put together at the request of a number of members of Congress who were concerned about the rising drumbeat of reports that the U.S. was losing ground in its race to stay in a dominant position in an increasingly competitive world. The report was fast-tracked and delivered in October 2005 after just six months of preparation, drawing on dozens of previous reports on the subject. It concluded that the U.S. was indeed at risk of losing its leadership role and recommended actions in three key areas: increasing research investment in the physical sciences; strengthening science, technology, engineering and

Balancing these interests is a political calculus, not a scientific one. In developing it, scientific information is sometimes pivotal, sometimes sits on the sidelines, and sometimes ends up as a mix where compromise may or may not embrace a technical truth.

mathematics education; and developing a more robust innovation infrastructure.


The report caught the attention of both the scientific community, which used it as a rallying point to advocate for increased funding, and the policy-making community who saw its clear statements about the threats the country faced if the current trends were not reversed. Policymakers were quick to draft responses to the document. The White House introduced a new presidential initiative called the American Competitiveness Initiative, modeled on some of the key recommendations of the Gathering Storm report, including doubling the research budgets of NSF, NIST, and DOE Office of Science over 10 years. The House Democratic leadership proposed an Innovation Agenda that echoed the report's recommendations. By mid-2006, legislation designed to enact the report's recommendations began to appear. Ultimately, these responses would coalesce into new legislation called the COMPETES Act that garnered unanimous approval in the Senate and overwhelming support in the House.

But this act merely laid out funding goals. Congress and the president still had to fund the various agencies as part of the annual appropriations process. The groundswell of support that was motivated by a clear and compelling rationale provided by the Gathering Storm report and the dozens of reports that had preceded it made it appear that Congress and the president would deliver on these promises. And, for most of the process, that was indeed the case. At every milestone during the FY 2008 funding cycle, the increases for NSF, NIST and DOE Office of Science were at or, in some cases, higher than the levels called for in the COMPETES Act. However, at the 11th hour, politics trumped the goals of the COMPETES Act.

The president and Republicans in Congress decided it was in their interest to constrain spending while Democrats wanted to increase spending for many of their priorities. Because of this fight between the president and the Democratic leadership—a fight the Democrats would ultimately lose because they could not override the president's veto—all non-defense spending, including all the science funding called for in COMPETES, was rolled into one giant omnibus spend-

It is difficult to predict what factors will motivate specific policy debates because the political system is ever changing.

ing bill. In order to get the funding levels to a level the president would sign, the Democratic leadership had to pick and choose which programs to grant priority and which to abandon. In need of a political victory in the wake of the defeat on the spending level, the Democratic leadership emphasized priorities with which they could draw the sharpest distinctions between their view and the president's. Unfortunately for the science community, that did not include a priority for science funding (for which, after all, the president shared priority). And so, despite having a strong case buttressed by numerous science advisory bodies and widespread support among policymakers, funding for those three key science agencies actually decreased in FY 2008 relative to inflation.

Our point is not to disparage those who would strive to ensure Congress and the administration act on strong technical and scientific grounds when crafting policy. Indeed, that is what both of our organizations ask us to do in Washington, D.C. Rather, it is to temper the inevitable frustration that has and will occur when Congress appears to act irrationally in its science and technology policy as it seeks to balance competing interests. As long as the current political incentives are in place, reviving OTA won't suddenly make Congress appear a great deal smarter about technology. 

Cameron Wilson (wilson_c@hq.acm.org) is the director of the ACM U.S. Public Policy Office in Washington, D.C.

Peter Harsha (harsha@cra.org) is the director of government affairs at the Computing Research Association (CRA) in Washington, D.C.

Calendar of Events

September 16–17

Softvis '08: International Symposium on Software Visualization, Munich, Germany, Contact: Christopher D. Hundhausen, Email: hundhaus@hawaii.edu

September 16–19

ACM Symposium on Document Engineering, Brazil, Contact: Maria da Graca Campos Pimentel, Phone: 55-16-3373-9657, Email: mgp@icmc.usp.br

September 16–19

ECCE08: European Conference on Cognitive Ergonomics, Madeira, Portugal, Contact: Joaquim A. Jorge, Phone: 351-21-3100363, Email: jaj@inesc.pt

September 20–23

The 10th International Conference on Ubiquitous Computing, Seoul, South Korea, Contact: Joseph McCarthy, Phone: 650-804-698, Email: joe@interrelativity.com

September 22–23

Multimedia and Security Workshop, Oxford, United Kingdom, Sponsored: SIGMM, Contact: Andrew David Ker, Phone: +44 1865 276602, Email: adk@comblab.ox.ac.uk

September 28–October 2

ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems (formerly UML), Toulouse, France, Sponsored: SIGSOFT, Contact: Jean-Michel Bruel, Phone: +33 686 002 902, Email: bruel@univ-pau.fr

September 29–October 1

Grid '08: 2008 IEEE/ACM International Conference on Grid Computing, Tokyo, Japan, Sponsored: SIGARCH, Contact: Stephanie Smith, Email: smith_s@acm.org

Viewpoint

Global Warming Toward Open Educational Resources

Seeking to realize the potential for significantly improving and advancing the world's standard of education.

THERE IS A looming discontent in the education world. Karen has dropped out of community college because her textbook costs exceeded her tuition bill. Eric, a third grader, must share his math textbook with his classmate because there aren't enough textbooks for all of the students. Juan's parents can't help him with his homework because they don't read English. Kelly, a science teacher, wonders whether Pluto will be reinstated as a planet by the time it is removed from her school's science textbooks. Rashid, a master teacher, is examining some of the 109,263 errors recently found in textbooks under review by the Texas State Board of Education. Patrick, a premedical student, is struggling to understand Newton's laws of motion from the text, formulas, and pictures in his textbook. Carla, an elementary school teacher, must purchase music materials out-of-pocket for her fourth-grade class due to a reduced school budget. And John, a university professor, is astonished to learn that the book he published three years ago is already out of print.

The buzz surrounding the high cost, limited access, static nature, and often low quality of the world's textbooks has reached a crescendo lately, with many claiming a serious threat to the future of the next generation, the training of work forces worldwide, and the democratic process in society. The current predicament lowers the quality of education in the developed world; even worse, it puts

education out of reach for many in the developing world.

Imagine another world that has forestalled this crisis. A world where textbooks and other learning materials are free for all on the Web, available in low-cost printed versions, adapted to many backgrounds and learning styles, interactive and immersive, translated into myriad languages, continually up to date and corrected, and never out of print. Imagine virtual labs that can be used any hour of the day (or night). While this world was just a dream a decade ago, the Open Educational Resources (OER) movement that aims to create it has begun to coalesce and gather momentum.

Enter Open Educational Resources

The OER movement is based on a set of intuitions shared by a wide range of academics: knowledge should be free

and open to use and reuse; collaboration should be easier, not more difficult; people should receive credit and kudos for contributing to education and research; and concepts and ideas are linked in unusual and surprising ways, not necessarily the simple linear forms that today's textbooks present. OERs promise to fundamentally change the way authors, instructors, and students interact worldwide.^{1,3,4}

The OER movement takes the inspiration of the open source software movement, mixes in the powerful communication and visualization abilities of the Internet and the Web, and applies the result to teaching and learning materials like course notes, curricula, labs, and textbooks. OERs include text, images, audio, video, interactive simulations, problems and answers, and games that are free to use and reuse in new ways by

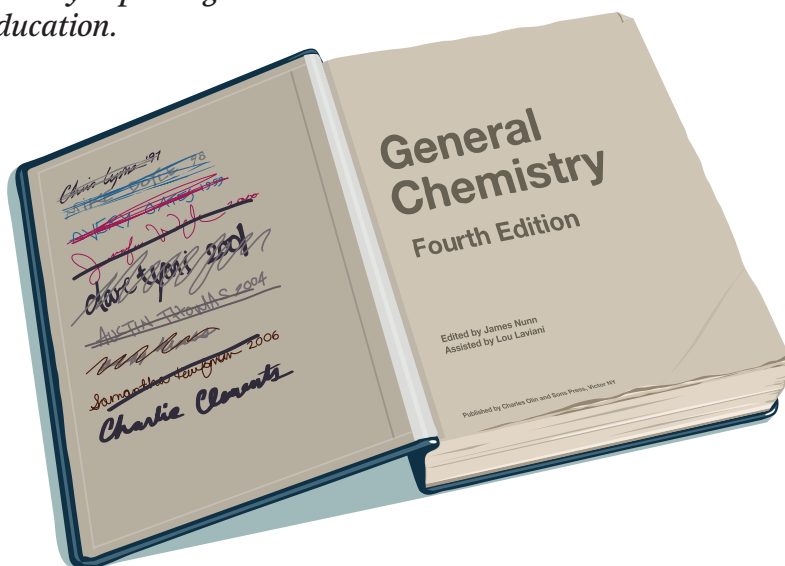


ILLUSTRATION BY CHRIS LYONS

anyone around the world.

Over the last decade the technical, legal, and social puzzle pieces have come together so that anyone, anywhere can now author, assemble, customize, distribute, have reviewed, and publish their own textbook in very little time and at zero or very low cost. The key enablers are:

- ▶ Technologies like the Internet, which enables virtually free digital content distribution; XML, which turns a monolithic textbook into a rapidly reconfigurable construction of small, reusable “modules,” much as building with Lego blocks; Web 2.0 tools like wikis and semantic tagging systems, which enable real-time distributed global collaboration; advanced visualization and graphics tools, which enable immersive simulation environments; and print-on-demand systems, which enable the production of inexpensive paper books for those who prefer or need them.

- ▶ Open copyright licenses like the Creative Commons and GNU Free Documentation licenses, which turn once closed and static educational materials into living objects that can be continuously developed, remixed, and maintained by a worldwide community of authors and editors.

Several OER projects are already attracting millions of users per month (as of July 2008). Some, like the MIT OpenCourseWare project (mit.edu/ocw) and its OCW consortium (ocwconsortium.org), are top-down organized institutional repositories that showcase their institutions' curricula. Others, like Connexions (cnx.org), are grassroots organized and encourage contributions from all comers. Still others, like the Open University's OpenLearn project (openlearn.open.ac.uk), combine aspects of both. Wikipedia (wikipedia.org) is regularly referenced by students, teachers, and faculty and is increasingly used directly as a learning tool. A consortium of community colleges throughout California and around the U.S. is developing a suite of free, open textbooks.² Governments like Vietnam's are committing to OERs to help reinvent their educational systems (vocw.vn). Professional societies like the IEEE are getting involved as a way to bolster their global educational outreach (ieeecnx.org). And the Student PIRGS Open Textbooks campaign (maketextbooksaffordable.org) is work-

ing to raise awareness of both textbook costs and this new avenue to reduce them. As a sign of the maturation of the movement, delegates from around the world met in Cape Town, South Africa to develop the eponymous Declaration that was officially released in January 2008 and has already garnered signatures from more than 1,600 individuals and 165 organizations to date (see cape-towndeclaration.org).

Free and Open are Just the Beginning

The most exciting thing about OERs is that free access is just the beginning. OERs will increasingly blur the lines between courses, grade levels, labs, and textbooks, turning the current textbook production pipeline into a vast dynamic knowledge ecosystem that is in a constant state of creation, use, reuse, and improvement. OERs also promise to provide each child with his or her own textbook that's tailored to the student's background and learning style (not “off the rack” as they are today) and to the institution's goals.

OERs enable the development of tighter feedback loops that immerse students in interactive learning environments and couple learning outcomes more directly into textbook development and improvement. A key online ingredient will be “Web 3.0/Semantic Web” technologies based on natural language processing, data mining, machine learning, artificial intelligence, and semantic markup languages like MathML, MusicXML, and CML (Chemical Markup Language). The result will be “textbooks” that not only deliver open content to students but also moni-

The buzz surrounding the high cost, limited access, static nature, and often low quality of the world's textbooks has reached a crescendo.

tor their interactions with them, analyze those interactions, and then send rich feedback to the student about their learning, as well as to the communities of curriculum builders, authors, and instructors to drive iterative improvement of the learning materials. An early example that currently focuses more on student feedback than continuous iterative content improvement is Carnegie Mellon University's Open Learning Initiative (cmu.edu/oli).

Free and Open as a Business Model

OERs are not at odds with the for-profit world. Indeed, we contend that the new development and distribution models promoted by the OER movement represent the natural and inevitable evolution of the educational publishing industry in a way that parallels the evolution of the software industry (the now-mainstream Linux, Apache, and Firefox), the music industry (Radiohead's recent “pay what you like” digital album download), and the scholarly publishing industry (the U.S. government's recent mandating of free online access to all journal articles stemming from NIH-funded research). The key enabler in all of these is free Internet-based digital distribution. Chris Anderson, in his *Wired* article “Free: Why \$0.00 is the Future of Business,” argues that while free was once a marketing gimmick, it is now emerging as a full-fledged economic model.

This economy provides many avenues for financially sustaining myriad different OER projects. Just as for-profit companies like Red Hat, IBM, Oracle, and others charge customers for the value they add to open source software and then in turn give back to the open source community through direct financial support, programming personnel, and free marketing, value-adding for-profit organizations are emerging in the OER space. For example, non-profit Connexions' partnership with for-profit QOOP (qoop.com) enables the production of print-on-demand paper textbooks that sell for a fraction of the price of a conventional commercial publisher (\$20 for a 300-page engineering textbook in regular use at Rice University; \$29 for a 500-page statistics textbook in use at a number of California community colleges starting in fall 2008). A three-way revenue-sharing arrangement benefits QOOP, Connexions, and the author (if

the author is interested in receiving royalties). And of course the content is also available for free on the Connexions Web site, which will keep the commercial costs from rising above what the value added justifies.

Roadblocks on the Horizon?

While the OER movement is rapidly gaining speed, there are a number of potential roadblocks that must be carefully navigated for it to prosper.

Technology fragmentation. If the OER community does not adopt common or compatible content and repository standards, then it risks fragmenting the movement into a number of isolated islands of incompatible content. This will unfortunately discourage global collaboration, reduce the overall economy of scale of the enterprise, and thus devalue any financial sustaining opportunities. We must pay attention to the lessons learned by groups like the World Wide Web Consortium and its standardization and maintenance of the HTML and XML standards.

Intellectual property fragmentation. Just as with open source software, there are a number of copyright licenses that can be applied to OERs. These various licenses present a number of compatibility issues. For instance, there is currently a debate regarding whether open materials should or should not be commercially usable. Licensing that renders open materials only noncommercially useable promises to protect contributors from potentially unfair commercial exploitation. A noncommercial license, however, not only limits the spread of knowledge by complicating the production of paper books, e-books, CDs and DVDs, but also cuts off potential future revenues that might sustain non-profit OER enterprises in the future. Interestingly, such an anticommercial stance is contrary to that of the more established open source software world, which greatly benefits from commercial involvement. Where would Linux and Apache be without the value-adding contributions of for-profit companies like Red Hat and IBM, for instance?

Quality control. How can OERs produced in a grass-roots fashion, by people with varying skill levels and degrees, for widely varied reasons, be adequately vetted for quality? The anxieties frequently aired about projects like Wikipedia

OERs have the potential to aid in the democratization of the world of knowledge.

and other open-authorship projects suggest they are threatened by the proliferation of massive amounts of low-quality material that might swamp the information environment and prove impossible to navigate. Traditional publishers, as well as institution-based OER projects like MIT OpenCourseWare, employ a careful internal review process before their content is made publicly available. However, such a pre-publication review cannot scale to keep up with the fast pace of community-based OER development, where materials may change daily or even hourly. Accept/reject decisions also create an exclusive rather than inclusive community culture. And finally, prereview does not support evaluation of modules and courses based on actual student learning outcomes. Some promising steps are being made in this direction. In one, Connexions recently rolled out a system of post-publication “lenses” that are open to an arbitrary number of third-party reviewers and editorial bodies. Several universities, companies, and professional societies are currently reviewing content for their lenses (see cnx.org/lenses).⁵

Success models. While the advantages of remixing and reusing educational content are readily apparent (and while authors already consciously and unconsciously remix ideas from myriad different sources as they compose), we need more OER success models to build upon. We surmise that the lack of a large number of models is due in a large part to technological barriers (which are gradually being overcome) and in a lesser part to several hundred years of academic community dynamics (which are being addressed by community-organized OER projects like the IEEE’s mentioned earlier).

Moving Forward

Our experience with OERs over the past eight years has convinced us that the movement has real potential to enable a revolutionary advancement of the world’s standard of education at all levels. Moreover, as it grows and spreads, the movement will have a large impact on the academic world itself. It promises to disintermediate the scholarly publishing industry, in the process rendering some current business models unviable and inventing new viable ones. It will also change the way we conceive of and pursue authorship, teaching, peer review, promotion, and tenure. And by encouraging contributions from anyone, anywhere, OERs have the potential to aid in the democratization of the world of knowledge.

A concerted effort from the community of authors, instructors, students, and software developers (that is, by you) will enable the OER movement to surmount the challenges on the road to these goals. Fortunately, it’s easy to get involved: become an author for an OER project on your favorite topic; contribute your out-of-print work so others can build on it and keep it alive; adopt or remix an open textbook for your next course; start or participate in an OER quality review program; or translate an OER into a new language. Together, we can change the way the world develops, disseminates, and uses knowledge. **□**

References

1. Baraniuk, R.G. Challenges and opportunities for the open education movement: A Connexions case study. Chapter in *Opening Up Education: The Collective Advancement of Education through Open Technology, Open Content, and Open Knowledge*, MIT Press, 2008.
2. “Bookless” textbook study launched at Foothill,” Palo Alto Online (May 5, 2008).
3. Breck, J. Editor of the special edition on open education in *Educational Technology* 47 (Nov.–Dec. 2007), 3–5; C. Sidney Burrus, “Connexions: An Open Educational Resource for the 21st Century,” 19–23; Stephen Carson, “The OpenCourseWare Model: High-Impact Open Educational Content,” 23–26.
4. Kelly, C.M. *Two Bits: The Cultural Significance of Free Software*. Duke University Press, 2008.
5. Kelly, C.M., Burrus, C.S., and Baraniuk, R.G. Peer review anew: Three principles and a case study in postpublication quality assurance. In *Proceedings of the IEEE* 96, 6 (June 2008).

Richard Baraniuk (richb@rice.edu) is the Victor E. Cameron Professor in the Electrical and Computer Engineering department at Rice University in Houston, TX.

C. Sidney Burrus (csb@rice.edu) is the Maxfield & Oshman Professor Emeritus of Engineering in the Electrical and Computer Engineering department at Rice University in Houston, TX.

The authors and Connexions are supported by the William and Flora Hewlett Foundation, the U.S. National Science Foundation, and Rice University.

essays, {craft, art, science} of software, python, eclipse, agile development, onward!, {generative, functional} programming, .net, open source, concurrency, smalltalk, aspects, second life, ruby, service-orientation, objects, embedded, ultra large scale {model, test}-driven passion, fun!, agents, domain-specific use cases, movies, lightning talks,



systems, objective-c, development, c#, design patterns, languages, wiki, product-lines, java, refactoring, plop

DEFINE THE FUTURE OF SOFTWARE

www.oopsla.org/submit

CONFERENCE CHAIR

GAIL E. HARRIS

Instantiated Software Inc.
chair@oopsla.org

PROGRAM CHAIR

GREGOR KICZALES

University of British Columbia
papers@oopsla.org

ONWARD! CHAIR

DIRK RIEHLE

SAP Research
onward@oopsla.org

CALL FOR PAPERS

March 19, 2008

Due date for Research Program, Onward!, Development Program, Educators' Symposium, Essays and proposals for Tutorials, Panels, Workshops and DesignFest

July 2, 2008

Due date for Development Program Briefs, Doctoral Symposium and Student Volunteers



Association for
Computing Machinery

NASHVILLE CONVENTION CENTER, NASHVILLE, TN
October 19 - 23, 2008

A study of the technology and sociology of Web service specifications.**BY IAN FOSTER, SAVAS PARASTATIDIS,
PAUL WATSON, AND MARK MCKEOWN**

How Do I Model State? Let Me Count the Ways

THERE IS NOTHING like a disagreement concerning an arcane technical matter to bring out the best (and worst) in software architects and developers. As every reader knows from experience, it can be hard to get to the bottom of what exactly is being debated. One reason for this lack of clarity is often that different people care about different aspects of the problem. In the absence of agreement concerning the problem, it can be difficult to reach an agreement about the solutions.

In this article we discuss a technical matter that has spurred vigorous debate in recent years: How to define interactions among Web services to support operations on state (that is, data values associated with a service that persist across interactions, so that the result of one operation can depend on prior ones.⁴ An airline reservation system and a scheduler

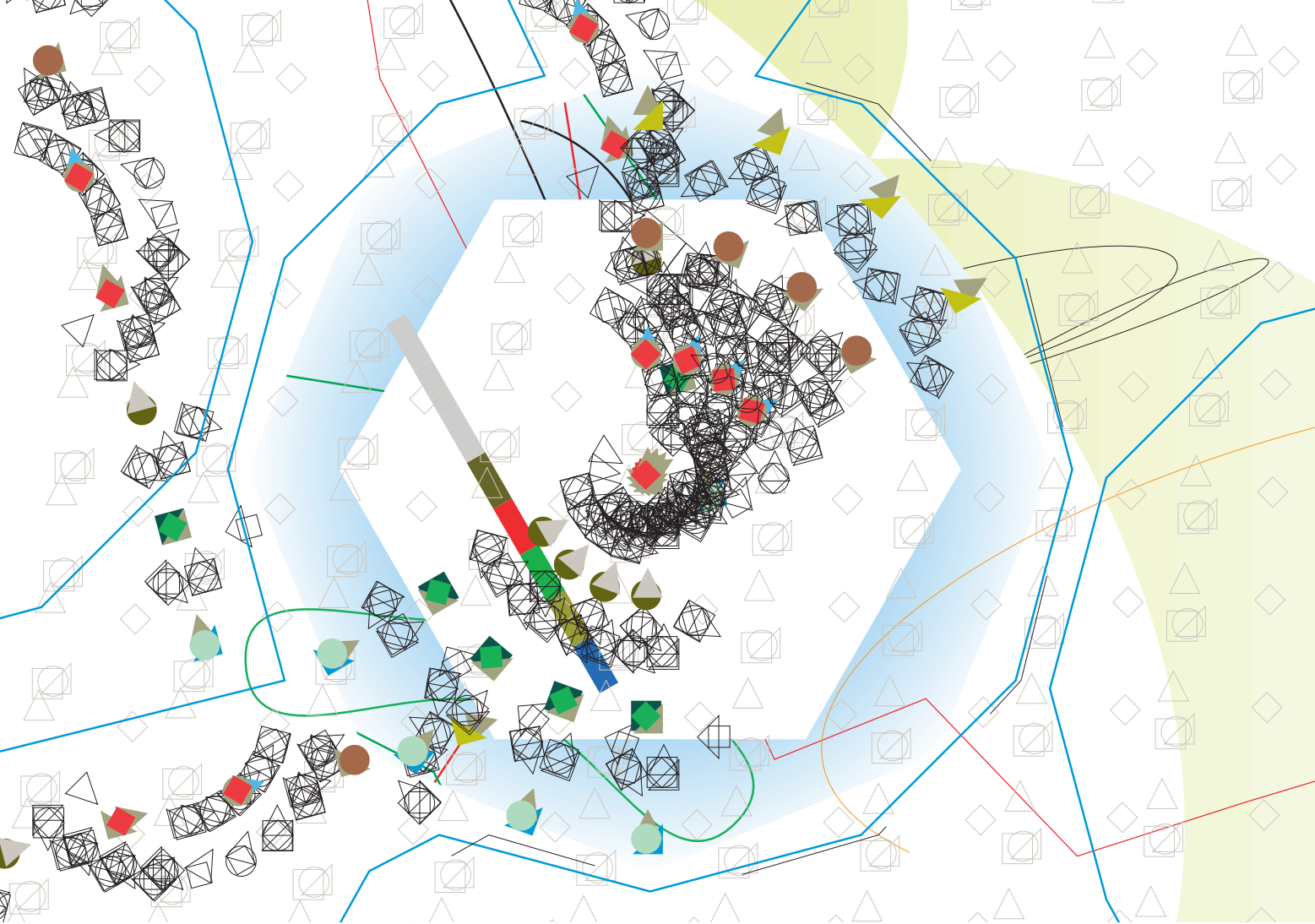
of computational jobs are two examples of systems with this requirement. Both must provide their clients with access to information about ongoing activities: reservations and jobs, respectively. Clients typically want to name and/or identify state (for example, refer to a specific reservation or job), access that state (get the status of a flight reservation or the execution progress of a job), modify part of that state (for example, change the departure time of a flight or set the CPU requirements of a job), and destroy it (for example, cancel a reservation or kill a job).

The debate over this issue does not concern the need for such operations but rather the specifics of how exactly to model and implement service state and the associated interactions on that state. For example, state may be modeled *explicitly* by the distributed computing technology used (for example, as an “object” with create, read, update, and destroy operations) or *implicitly* by referring to application domain-specific concepts within the interactions (for example, “create reservation,” “update reservation,” messages that include a domain-specific identifier like an Amazon ASIN in the body). Along a different dimension, we may use HTTP or SOAP as an implementation technology.

Our goal here is to shed light on possible approaches to modeling state. To this end, we present four different approaches and show how each can be used to enable access to a simple job management system. Then we summarize the key arguments that have been made for and against each approach. In addition to providing insights into the advantages and disadvantages of the different approaches, the discussion may also be interesting as a case study in technical debate. As we will see, the four approaches are remarkably similar in terms of what they do, but differ in terms of precisely how they do it.

Some Preliminary Observations

First, a few observations about what we mean by *modeling state*. The systems with which we want to interact



may have simple or complex internal state. Various aspects of this state may be exposed so that external clients can engage in “management” operations. For example, an airline reservation system might give customers the ability to programmatically create, monitor, and manage reservations. The same system might also allow operators to programmatically access information about current system load and the mapping of computational resources to different system functions. We are not suggesting these mechanisms provide direct access to the underlying state in its entirety. Rather, we are assuming the principles of encapsulation and data integrity/ownership are maintained. It is up to a system’s designer to define the projections to those aspects of the system’s internal state that they are willing to expose to the outside world.⁵

Such projections can be complex. For example, in the case of a job management system, the underlying state associated with even an apparently simple job may consist of multiple distinct processes on different back-end

computers, entries in various internal tables and catalogs, and activities within subsystems such as schedulers and monitors. When designing the allowed interactions with such a system, we must model the “state of a job” (the projection of the complex underlying state that is to be made available to clients) in a manner that is not only easy for clients to understand and use but that also makes it possible to maintain this projection effectively.

It is unwieldy to keep talking about “modeling a projection of underlying system state,” so we use the shorthand “modeling state.” It is important to bear in mind, however, the reality of what could be going on behind the boundaries of a system with which an interaction takes place.

We also make a few remarks concerning the difference between architectural styles and implementation technologies. The evolution of the Web from an infrastructure that enables access to resources to a platform for distributed applications has resulted in much discussion on the relevant ar-

chitectural approaches and technologies. Terms such as representational state transfer (REST;³ an architectural style) and HTTP (a protocol specification) are often used interchangeably to indicate an architectural approach in which a small set of verbs/operations (PUT, GET, DELETE) with uniform semantics are used to build applications. Similarly, the popularity of Web services (a set of protocol specifications)¹ has resulted in the use of that term as a synonym for service orientation (an architectural style).

We draw a distinction between the architectural styles and their implementation technologies. Instances of the former represent a collection of principles and constraints that provide guidance when designing and implementing distributed applications. In contrast, the latter are the mechanisms or tools used to apply the principles of an architectural style when building applications. There is not a one-to-one mapping between an architectural style and an implementation technology, even though one set of tools may

be easier to use when applying a particular set of principles. For example, pure HTTP is particularly well suited for implementing distributed applications according to REST principles, while Web services technologies such as SOAP are better suited for interface-driven applications. There is no reason, however, why one could not build a REST-oriented application using Web services technologies or a distributed object-based application using HTTP—although we doubt anyone would want to go through such an exercise.

Four Approaches to Modeling State

Table 1 summarizes the key properties of the four approaches presented here. The following provides a brief description of each approach.

The Web Services Resource Frame-

work (WS-RF) defines conventions on how state is modeled and managed using Web services technologies. WS-RF implements an architectural style similar to that of distributed objects or resources. Projected state is explicitly modeled as an XML document (the state representation) and is addressable via a WS-Addressing endpoint reference (EPR), a conventional representation of the information that a client needs to access a network service.

As in traditional object-based systems, any number of operations can be defined that access, or result in the change of, the projected state. The WS-RF specifications, however, define a set of common operations for the following: accessing that projected state (the XML document) in its entirety or in part; requesting notification of changes

on it (using WS-Notification); updating it in its entirety or in part; and deleting it. The structure of the XML document (that is, the schema), together with all the operations that can be applied to the projected state, known as the resource, are included in the Web Services Description Language (WSDL) document associated with the state's EPR, thus allowing clients to discover, using standard operations, what state a particular service makes available.

The WS-RF and WS-Notification specifications were developed within the OASIS standards organization. They are implemented within various open source and proprietary systems. Other specifications, notably WS-Notification and Web Services Distributed Management (WSDM), build on WS-RF.

Table 1: Key characteristics of the four approaches. In each box, we list conventional encodings of a function in terms of operation name(s) and (in brackets) the defining specification. The absence of an entry simply means that no conventional encoding has been defined; a custom, application-specific encoding can still be provided.

	WS-RF	WS-Transfer	HTTP	No conventions
State representation schema	WSDL extensions			
Address state representation	EPR (WS-Addressing)	EPR (WS-Addressing)	URI	URN
Create new state		Create (WS-Transfer)	HTTP POST	
Access entire state	GetResourcePropertyDocument (WS-ResourceProperties)	Get (WS-Transfer)	HTTP GET	
Get part of state	GetResourceProperty, GetMultipleResourceProperties, QueryResourceProperties (WS-ResourceProperties)		Not defined unless part of a state representation is exposed through a different URI (no semantics about the relationship are defined)	
Update entire state	SetResourceProperties (WS-ResourceProperties)	Put (WS-Transfer)	HTTP PUT	
Update, or add, part of state	SetResourceProperties, InsertResourceProperties, UpdateResourceProperties, DeleteResourceProperties (WS-ResourceProperties)			
Request notification	Subscribe (WS-Notification)	Subscribe (WS-Eventing)		Subscribe (WS-Eventing)
Lease-based lifetime management	SetTerminationTime (WS-ResourceLifetime)			
Destroy state	Destroy (WS-ResourceLifetime)	Delete (WS-Transfer)	HTTP DELETE	
Fault modeling	Well-defined error codes (WS-BaseFaults + other specs)	SOAP faults	HTTP fault codes	SOAP faults
RPC-based	Yes	Yes	No	No
Open standards process	Yes (OASIS)	Yes (W3C)	Already a standard	No need for new standards

WS-Transfer, like WS-RF, models the projected state explicitly through an XML document accessible via an EPR. However, the only operations defined on that state are, as per the CRUD (create, retrieve, update, and delete) architectural style: *create* a new resource state representation by supplying a new XML document; *get* an entire resource state representation; *put* a new resource state representation to replace an existing one; and *delete* an existing state representation. Distributed, resource-oriented applications are then built by using these operations to exchange state representations.

The WS-Transfer specification was developed by an industry group led by Microsoft and has recently been submitted to the World Wide Web Consortium (W3C) for standardization. Other specifications, notably WS-Eventing and WS-Management, build on WS-Transfer. As we will see later, WS-Transfer and WS-RF differ only in minor technical details; they arguably owe their separate existence more to industry politics than technical considerations. Fortunately, there seems to be industry support for an integration of the WS-Transfer and WS-RF approaches, based on a WS-Transfer substrate—the WS-ResourceTransfer specifications.

HTTP is an application protocol implementing a resource-oriented approach to building distributed systems. It has been described as an implementation of the REST architectural style. Like WS-RF and WS-Transfer, HTTP implements a resource-oriented approach to building distributed systems. According to REST, a small set of verbs/operations with uniform semantics should be used to build hypermedia applications, with the Web being an example of such an application. The constraints applied through the semantics of these operations aim to allow applications to scale (for example, through caching of state representations). State representations are identified through a URI. HTTP defines simple verbs—such as POST, PUT, GET, DELETE—and headers to enable the implementation of applications according to REST principles. XML is just one of the many media formats that HTTP can handle.

Finally, in the “no conventions for managing state” approach (“no-con-

Table 2: The eight operations considered in our comparison of different approaches.

#	Operation	Description
1	Create new job	A client requests the creation of a new job by sending a job creation request to a job factory service responsible for creating new jobs. Upon success, a job handle is returned that can be used to refer to the job in subsequent operations.
2	Retrieve state	Retrieve all state information associated with a specified job (e.g., execution status, resource allocation, program name).
3	Status	Determine the execution status (e.g., active, suspended) of a specified job.
4	Lifetime	Extend the lifetime of a specified job.
5	Subscribe	Request notification of changes in the state of a specified job.
6	Suspend	Suspend a specified job.
7	Terminate	Terminate a specified job.
8	Terminate multiple	Apply the terminate operation to all jobs that satisfy certain criteria, such as those belonging to a particular client, those with a particular total execution time, those with a specific current execution status, or those with explicitly identified job handles.

ventions” in the following),⁶ there are no such concepts as operations, interfaces, classes, state, clients, or servers. Instead, applications are built through the exchange of one-way messages between services. Semantics to the message exchanges (for example, whether a message can be cached or whether a transactional context is included) are added through composable protocols. State representations are not fundamental building blocks. Instead, resources should be identified through URIs (or URNs) inside the messages, leaving it up to the application domain-specific protocols to deal with state management. Although any asynchronous messaging technologies could be used in implementations following this style, we consider here an implementation based on Web services protocols, however, without the introduction of state-related conventions.

We use a simple example to provide a more concrete comparison of these four approaches. The example is a job management system that allows clients both to request the creation of computational tasks (“jobs”) and to monitor and control previously created jobs. It provides the eight operations listed in Table 2, which we choose to represent as a range of typical state manipulation operations. In each case, a client makes the request by sending an appropriate message to the job management system and then expects a response indi-

cating success or failure.

Operation 1 creates a new job and returns a *handle* that can be used to refer to the job in subsequent operations. Parameters specify such things as required resources, an initial lifetime for the job, and the program to be executed.

Operations 2–7 support some archetypal job monitoring and control functions on a single job.

Operation 8 is an example of an interaction that may relate to multiple jobs. The set of jobs to which the operation is to be applied might be specified either in terms of job characteristics or by supplying a set of job handles.

In the discussion that follows, we show how our four approaches can be used to build a service that supports these eight tasks. As we shall see, each approach not only has in common that the “job factory service” is a network endpoint to which job creation and certain other requests should be directed, but also is distinguished from the other approaches in terms of:

- Its *syntax* (that is, how the job handle should be represented and how operations on the job should be expressed in messages).

- Its use (or not) of *conventions* defined in existing specifications for the purpose of defining its syntax.

The distinction made here between syntax and conventions may appear unimportant, but we emphasize it so

Table 3: Syntax used in WS-RF job management interface, showing for each operation of Table 2 the request message, destination address, and return message.

#	Message	To	Returns on success
1	CreateJob(job-specification)	job-factory-service	WS-Resource-qualified EPR to job state (job handle)
2	GetResourcePropertyDocument()	job-handle EPR	XML document comprising all job state
3	GetResourceProperty ("status")	job-handle EPR	Job status
4	SetTerminationTime (lifetime)	job-handle EPR	New lifetime
5	Subscribe (condition)	job-handle EPR	EPR to subscription
6	Suspend	job-handle EPR	Acknowledgment
7	Destroy	job-handle EPR	Acknowledgment
8	DestroyMultiple(job-description)	job-factory-service	Acknowledgment

Table 4: Syntax used in WS-Transfer job management interface, showing for each operation of Table 2 the request message, destination address, and return message.

#	Message	To	Returns on success
1	Create (job-spec)	job-factory-service	EPR to job state
2	Get ()	job-handle EPR	XML document comprising all job state
3	Get ()	job-handle EPR	XML document comprising all job state, from which status can be extracted.
4	SetLifetime(lifetime)	job-handle EPR	New lifetime
5	Subscribe (condition)	job-handle EPR	EPR to subscription
6	Suspend	job-handle EPR	Acknowledgment
7	Delete ()	job-handle EPR	Acknowledgment
8	DeleteMultiple(job-spec)	job-factory-service	Acknowledgment

Table 5: Syntax used in REST job management interface, showing for each operation of Table 2 the request message, destination address, and return message.

#	Message	To	Returns on success
1	HTTP POST <i>job-specification</i>	job-factory-service (e.g., http://grid.org)	URI(s) identifying the job(s) that have been created: e.g., http://grid.org/Bloggs/Jobs/4523
2	HTTP GET	http://grid.org/Bloggs/Jobs/4523	A representation of the state of the job, including Xlinks and semantic information
3	HTTP GET	http://grid.org/Bloggs/Jobs/4523/status	A representation of the status of the job
4	HTTP PUT <i>exp-time</i>	http://grid.org/Bloggs/Jobs/4523/lifetime	Acknowledgment
5	HTTP POST <i>condition</i>	http://grid.org/Bloggs/Jobs/4523/subs	URI identifying the new subscription
6	HTTP PUT "suspended"	http://grid.org/Bloggs/Jobs/4523/status	Acknowledgment
7	HTTP DELETE	http://grid.org/Bloggs/Jobs/4523	Acknowledgment
8	—	—	—

that we can focus in this section on syntax and postpone discussion of the advantages or disadvantages of adopting specific “standards” (conventions) to later in this article.

WS-RF implementation. Table 3 describes a job management interface based on the WS-RF and WS-Notification specifications. Here, we use **bold-face type** to indicate operation names that are defined in some specification associated with the approach in question. Those operations that are not in boldface are, by definition, not defined in any existing specification, and thus their syntax and semantics represent somewhat arbitrary choices, selected for illustrative purposes. We see that WS-RF and WS-Notification specifications provide five of the eight required functions.

The job handle returned upon success from operation 1 is represented as an EPR. A client receiving such a job handle can then use it as a destination for operations 2–7. Note that requests are directed to the Web service address contained in the job handle EPR, which may or may not be the job factory service. This distinction is important because it allows for a logical and/or physical separation between the job factory and job management functions.

Operation 8 is sent directly to the job factory service, which is assumed to have access to information about all active jobs. The argument could be, for example, a specification (such as an Xpath specification) identifying the jobs that are to be terminated (for example, all jobs created by Bloggs or all jobs that have exceeded their quota, and/or a list of EPRs denoting the jobs to be terminated).

WS-Transfer implementation. Table 4 describes a job management interface based on the use of the WS-Transfer and WS-Eventing specifications, which provide five of the eight required operations.

As in the WS-RF interface, the job handle returned upon success from operation 1 is represented as an EPR, and a client receiving such a job handle can then use it as a destination for operations 2–7. Note that requests are directed to the Web service address contained in the job-handle structure, which may or may not be the job-factory service.

We note that an alternative treatment of operation 3 is possible, albeit with some extra work, that avoids the need to transfer the entire state document. A new operation (for example, GetEPRtoPart) is defined that requests that a new state representation be exposed, through a different EPR, representing parts of the original state representation. The Get() operation is then applied to this new EPR. WS-Transfer applications (and higher-level specifications such as WS-Management) often use this approach to address the lack of support for partial state access in WS-Transfer.


Operation 8 is sent directly to the job factory service, which is assumed to have access to information about all active jobs, as in the WS-RF case.

HTTP implementation. Table 5 summarizes the syntax of an HTTP implementation. Note that operation 5 can alternatively be addressed via some custom encoding or by using a system such as SMTP, Jabber, SMS, or Atom. HTTP DELETE cannot take any content, so there is no way to specify that a set of jobs (operation 8) can be deleted by using the HTTP DELETE message, except in the case when we delete all jobs in some predefined group (for example, “HTTP DELETE http://grid.org/Bloggs/Jobs” to delete all jobs created by Bloggs).


Note that whereas HTTP defines all the verbs used, the structure of the URIs and the format and semantics of the documents exchanged in order to implement the job service’s operations are application specific. Thus, while the URIs appear to convey some semantic information based on their structure (for example, a /status at the end of a particular job URI may be interpreted by a human as the identifier of the status resource), this is an application-specific convention.

No-conventions/Web services implementation. Since in this approach we assume no defined state management conventions with widely agreed semantics, each application domain is expected to define interactions that meet its own requirements. Table 6 summarizes a potential implementation of the job management service in this style, using SOAP messaging.

Because of the nature of the example chosen, all operations are defined



It is unwieldy to keep talking about “modeling a projection of underlying system state,” so we use the shorthand “modeling state.” It is important to bear in mind, however, the reality of what could be going on behind the boundaries of a system with which an interaction takes place.



as request-response message exchanges. The CreateJob message exchange returns an identifier as the job handle. The returned job identifier may be a globally unique URI (for example, a URN) that can be accepted by multiple job services. Metadata about it (for example, the job services that “know” about it) may be discovered from registries. Examples of this approach to identifying resources include the Life Science Identifier (LSID), International Virtual Observatory Alliance (IVOA) identifier, and Amazon Standard Identification Number (ASIN). Thus, the job identifier may become a technology-independent handle that can also be used with other technologies (for example, a Jini or CORBA interface to the same service). A client receiving such a job handle can then pass it to the job management service.

Discussion

The four approaches do not differ greatly in terms of what they actually do. All send essentially the same messages, with the same content, across the network. For example, a request to destroy a particular job will in each case be directed to a network endpoint via an HTTP PUT, and will contain the name of the operation to be performed, plus some data indicating the job that should be destroyed. The approaches vary only in how these different components are included in a message, an issue that may have implications for how messages can be processed and routed but that has no impact on how services are implemented.

At the same time, there clearly are significant differences among the approaches in terms of, for example, their use of conventions, the underlying protocols, and the tooling available to support their use. We summarize here important arguments that have been made on these topics. The characterizations of the various positions are our own.

The value of convention. Proponents of the WS-RF and WS-Transfer approaches argue that creating, accessing, and managing state involve a set of common patterns that can usefully be captured in a set of specifications, thus simplifying the design, development, and maintenance of applications that use those patterns. For example, in the

Table 6: Syntax used in job handle approach, showing for each operation of Table 2 the request message, destination address, and return message.

#	Message	To	Returns on success
1	"New job" carrying the job specification	job-management-service	Identifier for the newly created job
2	"State" carrying job identifier(s)	any job service aware of the job identifier(s)	XML document with job state(s)
3	"Status" carrying job identifier(s)	any job service aware of the job identifier(s)	XML document with job status(s)
4	"New lifetime" carrying job identifier(s) and new lifetime(s)	any job service aware of the job identifier(s)	Acknowledgment*
5	"Subscription" carrying job identifier(s) and subscription information (for each)	any job service aware of the job identifier(s)	Acknowledgment*
6	"Suspension" carrying job identifier(s)	any job service aware of the job identifier(s)	Acknowledgment*
7	"Destroy request" carrying job identifier(s)	any job service aware of the job identifier(s)	Acknowledgment*
8	"Destroy request" carrying job identifier(s) and query expression	any job service aware of the job identifier(s)	Acknowledgment*

* May also silently accept the message and report on fault only. If proof of delivery is necessary, WS-ReliableMessaging can be used.

Alphabet and Specification Soup

Any discussion of Web services inevitably involves a plethora of acronyms and specification names. We list some of them here. To save space, we do not provide citations for individual specifications. These can easily be located online.

EPR (endpoint reference)	As defined in the WS-Addressing specification, a combination of Web services (WS) elements that define the address for a resource in a SOAP header.
SOAP	A protocol for exchanging XML-based messages over networks, normally using HTTP.
WSDL (Web Services Description Language)	An XML-based language that provides a model for describing Web services.
WS-Eventing	A specification that defines a protocol for Web services to subscribe to another Web service or to accept a subscription from another Web service.
WSDM (Web Services Distributed Management)	An OASIS-developed Web services architecture and set of specifications for managing distributed resources.
WS-ResourceTransfer	A proposed integration of WS-RF and WS-Transfer.
WS-RF (Web Services Resource Framework)	An OASIS-developed architecture and set of specifications for describing and accessing state in a Web service.
WS-Transfer	A specification that defines a protocol for the transfer of an XML representation of a WS-addressable resource, as well as for creating and deleting such resources.

case of our job management service, these proponents might observe the following:

► The creation and subsequent management of a job can naturally

be viewed as an instance of some general patterns (creation, access, subscription, lifetime management, and destruction of state associated with a job); and

► The encoding of the job-management interface in terms of those patterns simplifies both the design of the interface (as much of it is already provided in other specifications) and the explanation of that interface to others.

They may also point out that programmer productivity is enhanced by client tools and applications that are "WS-RF and/or WS-Transfer aware." For example, a registry or monitoring system can use WS-RF operations to access service state without any application-specific knowledge of that state's structure.²

In contrast, proponents of the no-conventions approach argue that the design of any particular interface (for example, one for job management) will typically be relatively simple, involve issues that are not captured by these conventional patterns, and not require all features included in the specifications that encode that pattern. Thus, one can achieve a simpler design by proceeding from first principles. For example, in the case of our job management service, while WS-RF and WS-Transfer provide us with a Destroy operation "for free," we still need to introduce a separate Suspend operation. Furthermore, the semantics of Destroy may be quite application specific. For example, a service implementer may decide to retain information about "destroyed" jobs (by changing their status to "destroyed") so that information about them can still be retrieved. In this case, the state would not be destroyed. Finally, WS-RF and WS-Transfer focus on interaction with single states (for example, DestroyMultiple had to be custom defined) and so may not provide useful conventions in cases where operating on multiple states is the common case; for example, all Amazon REST and Web services can consume multiple ASINs.

Ideally, we would like to evaluate the relative merits of these two positions in terms of concrete metrics such as code size. Such an evaluation, however, requires agreement on the requirements that the interfaces should support. Unfortunately, proponents of the different approaches tend to differ also in their views of requirements. For example, a proponent of common patterns might see the ability to use WS-ResourceLifetime operations for soft-state lifetime management as de-

sirable, while others might not see that feature as important.

Standards. Another topic of disagreement concerns the importance of standardized specifications. Unfortunately (but not uncommonly in the world of Web services), we are faced with not one but two sets of Web services specifications, similar in purpose and design but different in minor aspects of syntax and semantics. WS-RF has been submitted to, and reviewed and approved by, a standards body (OASIS); WS-Transfer has been submitted to W3C but is not yet a W3C recommendation. The proposed consolidation of the two, WS-ResourceTransfer, adds to the confusion. Thus, we get a mix of opinions, including the following:

- ▶ WS-RF, having undergone intensive community review by a standards body, is therefore technically superior and/or morally preferable to the “proprietary” WS-Transfer.

- ▶ WS-RF is superior to WS-Transfer for technical reasons—for example, its support for access to subsets of a resource’s state, which can be important if that state is large. (In WS-Transfer, the same effect can be achieved, but only by defining an auxiliary operation that returns an EPR to a desired subset.)

- ▶ WS-Transfer is superior to WS-RF for technical reasons—for example, its greater simplicity.

- ▶ As a general principle, we should employ only specifications that are stable, widely accepted, and supported by interoperable tooling. Because neither WS-RF nor WS-Transfer is supported by all major vendors, neither passes this test. This view argues for the no-conventions approach.

- ▶ HTTP is widely used on the Web and, as a result, it should be preferred over any WS-based solution.

Implementation reuse. Proponents of conventions such as WS-RF and WS-Transfer argue that the adoption of conventional patterns can facilitate code reuse. Every WS-RF or WS-Transfer service performs such things as state access, lifetime management, concurrency control for incoming requests, and state activation/deactivation in the same way. Thus, the code that implements those behaviors can be reused in service implementations.

Proponents of the no-conventions

approach, however, reply that service implementers can also use the same code. In other words, there is certainly value to providing service implementers with standardized implementations of common tasks, but this need not imply that those patterns are exposed outside the service.

Simplicity vs. structure. Proponents of the HTTP/REST approach emphasize that it provides for more concise requests and permits the use of simpler client tooling than approaches based on Web services. Critics point out that the use of HTTP/REST means that users cannot leverage the significant investment in Web services technologies and platforms for message-based interactions.

Summary


We have presented four different approaches to modeling state in (Web) service interactions. Each approach defines roughly comparable constructs for referring to, accessing, and managing state components, but differs according to both its precise syntax and the use made (or not) of conventional domain-independent encodings of operations.

Thus, when defining state management operations, the WS-RF and WS-Transfer approaches both use EPRs to refer to state components and to adopt conventions defined in the WS-RF and related specifications and in the WS-Transfer and related specifications, respectively. In contrast, the no-conventions and REST approaches adopt domain-specific encodings of operations, on top of SOAP and HTTP, respectively.

Analysis of the debates that have occurred around these different approaches emphasizes the difficulties inherent in separating technical, political, and stylistic concerns. Some differences of opinion relate to well-defined technical issues and reflect either different philosophies concerning system design or different target applications. Others relate to differing target time scales. For example, no-conventions proponents initially argued against the use of WS-Addressing because of lack of support for that specification in certain tools, while WS-RF and WS-Transfer proponents argued in favor, believing that WS-Addressing would

eventually become universal. Support for WS-Addressing has since become quasi-universal, and now few find its use objectionable.

Acknowledgments

We are grateful to Karl Czajkowski, Jim Gray, and Sam Meder for comments on an earlier version of this document. Needless to say, the characterizations of the different arguments are our own. The work of the first author was supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. The work at Newcastle was supported by the UK eScience Programme, with funding from the EPSRC, DTI and JISC. 

References

1. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. and Orchard, D. *Web Services Architecture*. W3C, 2003.
2. Chervenak, A., Schopf, J.M., Pearlman, L., Su, M.-H., Bharathi, S., Cinquini, L., D’Arcy, M., Miller, N. and Bernholdt, D. Monitoring the Earth System Grid with MDS4. *2nd IEEE Intl. Conference on e-Science and Grid Computing*. Amsterdam, Netherlands, 2006.
3. Fielding, R. Architectural styles and the design of network-based software architectures. *Information and Computer Science*, University of California Irvine, 2000.
4. Foster, I., Czajkowski, K., Ferguson, D., Frey, J., Graham, S., Maguire, T., Snelling, D. and Tuecke, S. Modeling and managing state in distributed systems: The role of OGSI and WSRF. *Proceedings of the IEEE* 93,3, 604–612.
5. Helland, P. *Data on the Inside vs. Data on the Outside*. Microsoft Corporation, 2004.
6. Parastatidis, S., Webber, J., Watson, P. and Rischbeck, T. WS-GAF: A framework for building grid applications using Web Services. *Concurrency and Computation: Practice and Experience* 17, 2–4, 391–417.

Ian Foster (foster@anl.gov) is the director of the Computation Institute at Argonne National Laboratory, where he is also an Argonne Distinguished Fellow, and the University of Chicago, where he is also the Arthur Holly Compton Distinguished Service Professor of Computer Science.

Savas Parastatidis (Savas.Parastatidis@newcastle.ac.uk) is an architect in Microsoft Research. He investigates the use of technology in eResearch and is particularly interested in Cloud Computing, knowledge representation and management, and social networking.

Paul Watson (Paul.Watson@newcastle.ac.uk) is a professor of Computer Science at Newcastle University, and Director of the North East Regional e-Science Centre in the U.K.

Mark McKeown was a grid architect at the University of Manchester, U.K., at the time of this work.

© 2008 ACM 0001-0782/08/0900 \$5.00



DOI:10.1145/1378727.1378740

Smart power management is all about doing more with the resources we have.

BY MATTHEW GARRETT

Powering Down

POWER MANAGEMENT—FROM laptops to rooms full of servers—is a topic of interest to everyone. In the beginning there was the desktop computer. It ran at a fixed speed and consumed less power than the monitor it was plugged into. Where computers were portable, their sheer size and weight meant that you were more likely to be limited by physical strength than battery life. It was not a great time for power management.

Now consider the present. Laptops have increased in speed by more than 5,000 times. Battery capacity, sadly, has not. With hardware becoming increasingly mobile, however, users are demanding that battery life start matching the way they work. People want to work from cafés. Long-haul flights are now perceived as the ideal opportunity to finish a presentation. Two hours of battery life just isn't going to cut it; users are looking for upward of eight hours. What's drawing that power, and more importantly, how can we reduce it?

The processor is, perhaps, the most obvious target of power management. On a modern system the CPU is likely to be the single component consuming the most power. Switching the state of hundreds of millions of transistors takes power. Modern processors can generate well over 100 watts of heat. How can this be reduced?

At the most basic level the answer is simple: reduce the power taken to switch those transistors. Making them smaller is one mechanism, as is reducing the amount of power lost via leakage. There are limits, however, to how much power can be saved via fundamental improvements in semiconductor technology, and vendors are being forced to adopt increasingly high-tech solutions to maintain the rate of progress in this respect. We cannot rely on technological breakthroughs to be the silver bullet. We need to be smarter in how we use what we already have available.

The most reasonable approach to reducing the power consumption of processors is to realize that under normal usage patterns processors will not run at 100% utilization all the time. The first attempt to take advantage of this was in the Pentium era with the addition of idle power savings. The HLT (halt) instruction allowed operating systems to indicate that they had nothing to execute, letting the processor put itself into some form of power saving until the next interrupt arrived. Runtime processor power management had arrived.

The APM (Advanced Power Management) specification extended this functionality. Rather than simply halting the processor, the APM CPU idle call allowed the processor to be put into a lower power state where it continued to execute instructions. If system load reached a threshold level, the operating system could then send a CPU busy call to restore the processor to full operating speed.

This concept of runtime power management has been further enhanced in recent years. CPU clock and volt-

age scaling (such as Intel's SpeedStep) allows the clock rate of an idle processor to be reduced. Running more slowly provides greater tolerances for the processor, thus reducing the supply voltage. Reducing the clock speed provides a linear reduction in power usage; the simultaneous reduction of voltage allows the power consumption to be reduced quadratically.

As an orthogonal approach, modern processors implement dynamic power management by allowing the operating system to trigger entry into a range of low-power states when there is nothing to be executed. The most basic of these corresponds to the traditional behavior of the HLT instruction, but deeper sleep states allow the processor package to disable parts of itself. In the deepest states the processor can dissociate itself from the memory bus, disable much of the cache, and reach a state where it consumes a negligible quantity of power.

Moving between these low-power states takes time, with the deeper states taking longer to enter and leave. Since parts of the processor have been disabled, there's no way for code to be executed. The processor must be raised back to the full-power state before it can handle the next instruction.

As a result, deeper sleep states provide their greatest benefits only when the system will spend a significant period of time (20 milliseconds or more) truly idle. Most preemptive multitasking operating systems implement scheduling by allocating a time slice to running applications and forcibly switching to another task at the end of this time slice. Traditionally this time slice has been on the order of 10 milliseconds, meaning that an interrupt will be fired 100 times a second to enforce

scheduling. This interrupt is fired even if the system is idle, thus waking up the processor. A traditional fixed-tick operating system is therefore unable to obtain the greatest benefits from modern processor power savings.

A new model of scheduling was introduced in version 2.6.21 of the Linux kernel. Rather than having a static timer tick, the kernel looks for a list of tasks waiting to run. If the system is idle, it will look for the next expected timer expiry (such as an application that has requested to sleep for 200 milliseconds) and program the timer interrupt to fire at that point. The system will then sleep until either the next interrupt is fired or some external hardware interrupt occurs.

In the best-case scenario, this dynamic tick model allows the processor to be almost entirely shut down until the next time the user interacts with the computer. The real-world scenario is, unsurprisingly, worse. Many software programs set timers, and when these timers expire the processor must wake up to handle the interrupt.

These timers are often unnecessary. Some are simply polling loops that could be replaced by an interrupt-driven design. Others are straightforward design flaws—for example, an email client that checks local mailboxes for changes every 100 milliseconds even though the mail download interval is set to five minutes. Removing these loops can have a surprisingly large impact upon power consumption.

Sometimes, however, an application has no choice but to poll. In this case, it makes sense to consolidate as many loops as possible. Because of the latency required when switching into low-power states, a processor that wakes once a second and spends

four milliseconds executing code will actually draw less power than waking twice a second and spending two milliseconds executing code each time. The GLIB library used by GTK (GIMP toolkit) includes a helper function to make this easier. The `g_timeout_add_seconds()` function provides a time-out that will fire after a certain number of seconds, but only with second-level granularity. All time-outs scheduled to fire in a given second will be called at once, avoiding the need to add manual synchronization. Even better, this is synchronized across all GLIB-using applications. Using this in all applications on a system can significantly reduce the number of wake-ups per second, providing a drop in power consumption.

Of course, these issues are not limited to user-space code. Device drivers may have the same issues, and for many of the same reasons. The Linux kernel includes a function similar to GLIB's, allowing unavoidable polling to be synchronized. In some cases, however, the issues can be subtler. For example, hardware may continue sending events even when no information is present, unless it's told to stop. If the driver is well written, it will handle the case of empty data packets cleanly; thus, the absence of any support for quiescing the hardware may go unnoticed. Driver authors should disable hardware when it's not doing anything and avoid polling it if nobody is going to use the information.

The combination of processor-frequency scaling and idle-power states provides a somewhat surprising result. On almost all modern hardware, if there is code to run, then it is more power efficient to run the processor at full speed. This "race to idle" concept

How to Reduce Power Consumption

► **Hardware designers:** Make sure that your hardware never requires polling. Send interrupts on all status changes. Provide fine-grained control of logical subunits of the hardware in order to let the operating system power down as much as possible based on the functional

constraints imposed by the user.
► **Device driver authors:** Make sure that your driver doesn't poll. If nothing is listening to the driver (for example, if nothing is using a Webcam) then quiesce the hardware. Provide a rich interface to the device, allowing user space to

make it clear which functionality it requires. Leave the device in the lowest power state that satisfies these criteria.

► **Kernel developers:** Remove any dependency on a fixed clock. If you know that no tasks are waiting to run, don't wake up hun-

dreds of times a second to check the run queue.

► **Application authors:** Don't poll. Only run timer loops when absolutely necessarily and make sure they're synchronized. Don't keep hardware open if you don't need it.

stems from the power consumption of an idle processor being much lower than an active processor, even if running at a lower speed. The overall power consumption will be less if the processor spends a short time at full speed and then falls back to idle, rather than spending twice as long being active at a lower frequency.

Paying the Price for Graphics

Although the processor is the major component in determining power consumption, other parts of the platform also contribute. LCD panels in laptops are perhaps the most obvious secondary source. Modern hardware is moving from traditional cathode tube backlights to LED lighting, saving a small but significant quantity of power while providing the same level of illumination. Subtler techniques are beginning to appear in recent graphics chipsets. Intel has introduced technology to monitor the color distribution across the screen, which allows the backlight to be dimmed when the majority of the screen is dark. The intensity of brighter colors can be increased in order to compensate, resulting in power savings with little user-visible degradation.

Compressing the contents of the framebuffer can result in an additional reduction of power draw. A simple run-length encoding is often enough to achieve a significant reduction in image size, which means that the video hardware can reduce the amount of data that has to be read from the framebuffer memory and transferred to the screen. This simple optimization can save about 0.5 watts. An intriguing outcome of this is that desktop wallpaper design may influence system power consumption.

Hard drives have long been one of the more obvious aspects of power management. Even with all the advances made in recent years, hard-drive technology still requires an object to spin at high speed. In the absence of perpetual motion, this inevitably consumes power. For more than a decade operating systems have included support for spinning hard drives down after periods of inactivity. This indeed reduces power consumption, but it carries other costs. One is that disks are rated for only a certain number of



There are limits to how much power can be saved via fundamental improvements in semiconductor technology, and vendors are being forced to adopt high-tech solutions to maintain the rate of progress. We cannot rely on technological breakthroughs to be the silver bullet. We need to be smarter in how we use what we already have available.



spin-up/spin-down cycles and risk mechanical failure if this number is exceeded. A less-than-optimal approach to saving hard-drive power can therefore reduce hard-drive life expectancy.

Another cost is that spinning a disk back up uses more power than keeping a disk spinning. If access patterns are less than ideal, a drive will be spun up again shortly after being spun down. Not only will this result in higher average power consumption, but it will also lead to undesirable latency for applications trying to use the disk.

Making effective use of this form of power management requires a sensible approach to disk I/O. In the absence of cached data, a read from disk is inevitably going to result in spinning the disk back up. From a power management viewpoint, it makes more sense for an application to read all the data it is likely to need at startup and then avoid reading from disk in the future. Writes are more interesting. For most functionality, writes can be cached indefinitely. This avoids unnecessary disk spin-up, but increases the risk of data loss should the machine crash or run out of power.

The Linux kernel's so-called "laptop mode" offers a compromise approach in which writes are cached for a user-definable period of time if the disk is not spun up. They will be written out either when the user's time threshold is reached or when a read is forced to hit the disk. This reduces the average length of time that dirty data will remain cached, while also trying to avoid explicitly spinning the disk up to flush it.

Another way of avoiding write-outs is to reduce the number of writes in the first place. At the application level, it makes sense to collate writes as much as possible. Instead of writing data out piecemeal, applications should keep track of which information needs writing and then write it all whenever triggering any sort of write access.

At the operating-system level, writes can be reduced through careful consideration of the semantics of the file system. The metadata associated with a file on a Unix system includes a field that records the last time a file was accessed for any reason. This means that any read of a file will trigger a write to update the atime (time of last access),

even if the read is from the cache. One work-around is to disable atime updates entirely, but this breaks certain applications. A subtler work-around is to update the atime if the file was modified more recently than it was last read. This avoids breaking mail applications that depend upon the atime to determine whether mail has been read or not. This provides a dramatic reduction in write activity to the disk and makes it more likely that spinning the drive down will be worthwhile.

The drive is not the only part of the I/O system where savings are possible. The AHCI (Advanced Host Controller Interface) specification for serial ATA controllers includes link-level power management. In the absence of any pending commands, the link between the drive and the controller can be powered down, saving about 0.5 watts of power. The cost is a partial reduction in functionality—hotplug events will no longer be generated.

Network hardware provides a similar dilemma. If an Ethernet device is not in use, it makes sense to power it down. If the Ethernet PHY is powered down, however, there will be no interrupt generated if an Ethernet cable is plugged into the device. Power has been saved, but at the cost of some functionality. An almost identical problem occurs when detecting hot-plugging of monitors.

This is perhaps the most unfortunate side of power management. In many cases, hardware consumes power because that power is required to provide functionality. Many users may not care about that loss of functionality, but disabling it by default causes problems for those who do. Although hardware support for power management has improved hugely over recent years, the biggest challenge facing operating-system developers may in fact be how to integrate that support in a way that doesn't frustrate or confuse users.

In an attempt to encourage power management, Intel has released a tool called PowerTOP (referring to the top command used to see which processes are consuming the most CPU on Unix systems). PowerTOP uses diagnostic information from the Linux kernel to determine which applications are triggering CPU wake-ups, allowing

developers to spot misbehaving applications and optimize their power consumption.

PowerTOP has already provided benefits for Linux desktop software. The 7.04 release of Ubuntu ran at about 400 wake-ups a second. Optimizations and bug fixes in response to issues raised by PowerTOP allowed this to be reduced to fewer than 100 on most systems, with figures under 30 achievable on systems with disabled wireless and Bluetooth.

PowerTOP also provides information about other aspects of a system that may be consuming power. The system is probed to determine whether it has any configuration that may impair power savings, such as disabled USB autosuspend or audio codec power management, and PowerTOP suggests ways to fix the problem. The information provided is probably excessively technical for the average user, but it allows vendors to ensure they are taking full advantage of available power management options.


Though developed and sponsored by Intel, the advice provided by PowerTOP is fairly general and appropriate to most Linux-based systems. Arguably, the most important way in which it has been successful is not in the functionality it provides in itself, but the increased awareness of the issues involved that it has generated in the open source community. It remains to be seen whether proprietary vendors will start providing similar functionality to users and developers, but many of the same issues apply and need to be solved in similar ways.

Power Management for the Masses

The One Laptop Per Child XO machine is an interesting case study in power management, perhaps sharing more in common with the embedded world than with traditional laptops. Its aggressive power management is designed to allow the platform to suspend even when the machine is in use, something made possible by the display controller's ability to scan out the framebuffer even when the CPU isn't running. The mesh networking capability of the hardware requires machines to continue routing packets even when suspended, and hence the wireless hardware has also been de-

signed to forward data without processor intervention. Price considerations have meant that the XO machine cannot depend on the latest battery technology; therefore, the machine must consume as little power as possible to keep the network functioning effectively.

This level of power management would be unthinkable in the traditional laptop world, where even the best implementations still take on the order of a second to return to a state where the user can interact with the system. A major focus has therefore been to reduce the time taken to bring the device back from suspend, making it practical to suspend the device when idle without impairing the user experience. This requires a high level of robustness, and much of the development work has been focused on ensuring that components resume in a reliable and consistent manner. A failure rate of one in every 10,000 suspend/resume cycles might be considered acceptable in the mainstream laptop world, but would impair the user experience on the XO.

As users demand longer battery life and become increasingly concerned about wasted energy, power management has become more important to vendors. A well-rounded power management strategy requires integration of hardware, firmware, and software, as well as careful consideration of how to obtain the maximum savings without making the user aware of any compromised functionality. Future years are likely to see tighter integration and a greater awareness of good power management practices, and all-day computing may soon become a practical reality. 

Matthew Garrett (mjpg59@srcf.ucam.org) is a software engineer at Red Hat, specializing in power management and mobile hardware support.

A previous version of this article appeared in *ACM Queue*, Nov/Dec 2007, Vol. 5, No. 7.

Leaders in the storage industry ponder upcoming technologies and trends.

BY MACHE CREEGER, MODERATOR

CTO Storage Roundtable, Part Two

THE FOLLOWING CONVERSATION is the second installment of a CTO roundtable forum featuring seven world-class experts on storage technologies. This series of CTO forums focuses on the near-term challenges and opportunities facing the commercial computing community. Overseen by the ACM Professions Board, the goal of the series is to provide IT managers access to expert advice to help inform

their decisions when investing in new architectures and technologies.

Once again we'd like to thank Ellie Young, Executive Director of USENIX, who graciously invited us to hold our panel during the USENIX Conference on File and Storage Technologies (FAST '08) in San Jose, CA, Feb. 27, 2008. Ellie and her staff were extremely helpful in supporting us during the conference and all of us at ACM greatly appreciate their efforts. —*Stephen Bourne*

Participants

Steve Kleiman—Senior Vice President and Chief Scientist, Network Appliances.

Eric Brewer—Professor, Computer Science Division, University of California, Berkeley, Inktomi co-founder (acquired by Yahoo).

Erik Riedel—Head, Interfaces &

Architecture Department, Seagate Research, Seagate Technology.

Margo Seltzer—Herchel Smith Professor of Computer Science, Professor in the Division of Engineering and Applied Sciences, Harvard University, Sleepycat Software founder (acquired by Oracle Corporation), architect at Oracle Corporation.

Greg Ganger—Professor of Electrical and Computer Engineering, School of Computer Science, Director, Parallel Data Lab, Carnegie Mellon University.

Mary Baker—Research Scientist, HP Labs, Hewlett-Packard.

Kirk McKusick—Past president, Usenix Association, BSD and FreeBSD architect.

Moderator

Mache Creeger—Principal, Emergent Technology Associates.



IBM introduced the System/360 Model 30 with the tape drive system in 1965.

MACHE CREEGER: What can people who have to manage storage for a living take from this conversation? What recommendations can we make? What technologies do you see on the horizon that would help them?

STEVE KLEIMAN: Storage administrators today have tremendous problems that are not adequately solved by any tools. They have home directories, databases, LUNs. It's not just one set of bits on one set of drives; they're all over the place. They've got replicas and perhaps have to manage mirroring relationships between them. They have to manage a disaster recovery scenario and the server infrastructure on the other site if the whole thing fails. They have all these mechanisms for all these data sets that they must process day-in and day-out and they have to monitor the whole thing to see if it's working correctly. Just being able to manage that mess, the thousands of data sets they have to deal with, is a big problem that isn't solved yet.

MACHE CREEGER: Nobody's in the business of providing enterprise-level storage infrastructure management?

STEVE KLEIMAN: The people who have

solved it best in the past have been the backup people. They actually give you a data transfer mechanism that manages everything in the background and they give you a GUI that allows you to say, "I want to look for this particular data set, I want to see how many copies of it I have, and I want to restore that particular thing." Or "I want to know that these many copies have been made across this much time."

Of course, the problem is that it's all getting blown up. So now, it's not just, "What copies do I have on tape? What copies do I have in various locations spread around the world? What mirroring relationships do I have?" The trouble is that today it's all managed in someone's head. I call it "death by mirroring." It's hard. We'll sort it all out eventually.

KIRK MCKUSICK: What do you see as a possible solution?

STEVE KLEIMAN: Currently people are building outrageous ad hoc system scripts—Perl scripts and other types. My company is working on this as are lots of other people in the storage industry, but it's more than a single box problem. It's managing across boxes,

even managing heterogeneously. We have to understand that we're solving the convergence of QoS, replication, disaster recovery, archive, and backup. What we need is a unified UI for handling all these functions, each of which used to be handled for different reasons by different mechanisms.

ERIC BREWER: That is a core issue. How many copies do you have and why do you have them? Every copy is serving some purpose, whether as a backup, or a replication for read throughput, or a cache copy in flash. Because they're automatically distributed you can't keep track of all these things. I think you actually can manage the file system—broadly speaking, storage system, whereby you proactively assign how many copies you have of something.

MARGO SELTZER: Users make copies outside the scope of the storage administrator all the time.

ERIK RIEDEL: Because the amount of data and what it's used for both increase constantly, you have to get the machines to help the users tag content with metadata—to help them know what the data is, what the copy is for, where it came from, why they have it,

and what it actually represents.

MARGO SELTZER: With the data provenance you can identify copies, whether they were made intentionally or unintentionally. That's a start. However, answering the other semantic questions, such as "Why was the copy made?" will still require user intervention, which historically has been very difficult to get.

STEVE KLEIMAN: Each set of data—a database, a user's home directory—has certain properties associated with it. With a database you want to make sure it has a certain quality of service, a disaster recovery strategy, and a certain number of archival copies so that they can go back a number of years. They may also want to have a certain number of backup checkpoints to go back to in case of corruption.

Those are all properties of the data set that can be predefined. Once set, the system can do the right thing, including making as many copies as is relevant. It's not that people are making copies for the sake of making copies; they're trying to accomplish this higher-level goal and not telling the system what that goal is.

MARGO SELTZER: You're saying that you need provenance and you need the tools to add the provenance, so that when Photoshop makes a copy there's a record that says, "Okay, this is now a Photoshop document, but it came from this other document and then it was transformed by Photoshop."

ERIC BREWER: I completely agree with provenance, but I thought you said that it was inherently not going to work because users could always make copies that are not under anyone's control. I think that's the breach and not the observance. Most copies are made by software.

MARGO SELTZER: I agree, but I think that those copies have a way of leaking outside of the domain where things like de-duplication can't do anything about them. What typically happens is I go through the firewall, open up something on the corporate server, and then, as I am about to go on my trip, I save a file to my laptop and take my laptop away. Steve's de-duplication software is never going to see my laptop again.

ERIC BREWER: Yes, and that was my earlier point about managing the data. If you were to go to any system administrator with that scenario they'd get



STEVE KLEIMAN

Over the next decade enterprise-level data is going to migrate to a central archive function that is compressed and de-duplicated, potentially with compliance and whatever other disaster recovery features that you might want.



these big eyes and be really afraid. It should be a lot harder to do exactly what you just stated. That particular problem is perceived as a huge problem by lawyers and system administrators everywhere. The leakage of that data is a big issue.

STEVE KLEIMAN: Companies that actually own the end-user applications will have to set architectures and policies around this area. They'll certainly sign and possibly encrypt the document. Over time, they will also take responsibility for the things that we have been talking about: encryption, controlling usage, and external copies. Part of this problem is solved in the application universe and there are only a few companies that are practical owners of that space.

MARGO SELTZER: There are times when you want that kind of provenance and there are times when you really don't.

MACHE CREEGER: There's going to be a hazy line between the two. Defining what is an extraneous copy or derivation of a data object will be intimately tied up with the original object's semantics. Storage systems are going to be called on to have a more semantic understanding of the objects they store, and deciding that information is redundant and delete-able will be a much more complex decision.

STEVE KLEIMAN: The good news is the trend for end-user application companies, such as Microsoft, is to be relatively open about their protocols. Having those protocols open and accessible will allow people to leverage a common model across the entire system. So, yes, if you kept encrypting blindly you'd defeat any de-duplication because everything is Klingon poetry at that point. I should be able to determine whether two documents that are copied and separately encrypted are the same or not. I'm hoping that will be possible.

MACHE CREEGER: What recommendations are we going to be able to make? If IT managers are going to be making investments in archival types of solutions, disaster recovery, de-duplication, and so on, what should they be thinking about in terms of how they design their architectures today and in the next 18 months?

STEVE KLEIMAN: Over the next decade enterprise-level data is going to mi-

grate to a central archive function that is compressed and de-duplicated, potentially with compliance and whatever other disaster recovery features that you might want. Once data is in this archive and has certain known properties, the enterprise storage manager can control how it is accessed. They may have copies out on the edges of the network for performance reasons—maybe it's flash, maybe its high-performance disks, maybe it's something else—but for all that data there's a central access and control point.

MACHE CREEGER: So people should be looking at building a central archival store that has known properties. Then, once a centralized archive is in place, people can take advantage of other features, such as virtualization or de-duplication, and not sweat the peripheral/edge storage stuff as much.

STEVE KLEIMAN: I do that today at home, where I use a service that backs up all the data on my home servers to the Internet. When I tell them to back up all my Microsoft files, the Microsoft files don't go over the network. The service knows that they don't have to

copy Word.exe.

MARY BAKER: I'm going to disagree a little bit. One of the things I've been doing the last few years is looking at how people and organizations lose data. There's an amazing richness of ways in which you can lose stuff and a lot of the disaster stories were due to, even in a virtual sense, a centralized archive.

There's a lot to be said for having those edge copies under other administrative domains. The effectiveness of securing data in this way depends on how seriously you want to keep the data, for how long, and what kind of threat environment you have. The convenience and economics of a centralized archive are very compelling, but it depends on what kinds of risks you want to take with your data over how long a period of time.

MARGO SELTZER: What happens if Steve's Internet archive service goes out of business?

STEVE KLEIMAN: In my case, I still have a copy. I didn't mean to imply that the archive is in one location and that there's only one copy of that data in the archive. It's a distributed archive, which has bet-

ter replication properties because you want that higher long-term reliability. From the user's point, it's a cloud that you can pull documents out of.

ERIK RIEDEL: The general trend for the last several years is for more distribution, not less. People use a lot of high-capacity, portable devices of all sorts, such as BlackBerrys, portable USB devices, and laptops. For a system administrator, the ability to capture data is much more threatening today. Five or 10 years ago all you had to worry about were tightly controlled desktops. Today things are a great deal more complicated.

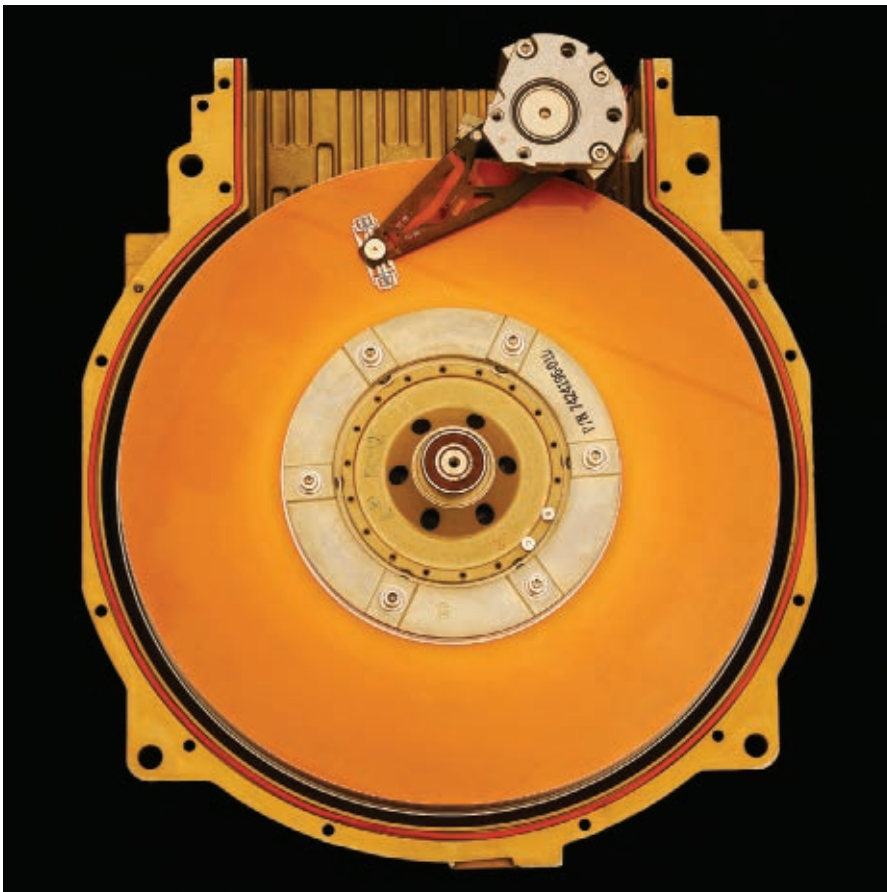
I was at a meeting where someone predicted that within two or three years, corporations were going to allow you to buy your own equipment. You'd buy your own laptop, bring it to work, and they'd add a little bit of software to it. But even in the age in which corporate IT departments control your laptop and desktop, certainly the train has left the station on BlackBerrys, USBs, and iPods. So for a significant segment of what the administrator is responsible for, pulling data back into a central store is not going to work.

MACHE CREEGER: That flies in the face of Steve's original argument.

STEVE KLEIMAN: I don't think so. I do think that there will be a lot of distributed data that will be on the laptops. There will be some control of that data, perhaps with DRM mechanisms. Remember, in an enterprise the family jewels are really two things: the bits on the disks and the brain cells in the people. Both are incredibly important and for the stuff that the enterprise owns, that it pays its employees to produce, it's going to want to make sure those bits exist in a secure place and not just on somebody's laptop. There may be a copy encrypted on somebody's laptop and the enterprise may have the key, but in order for the company to assert intellectual property rights on those bits, you are going to have to centrally manage and secure them in some way, shape, or form.

ERIC BREWER: I agree that's what corporations want, but the practice may be quite different.

STEVE KLEIMAN: That's the part I disagree with because part of the employee contract is that when they generate bits that are important to the company, the



The DEC Hard Drive Assembly, circa 1970.

company has to have a copy of them.

GREG GANGER: Let's be careful. There are two interrelated things going on here: Does the company have a copy of the information, and can a company control who else gets a copy? What Erik just brought up is an example of the latter. What Steve has been talking about is more of the former.

STEVE KLEIMAN: Margo has been saying that companies may not have a copy. I fundamentally disagree with that. That's what it pays the employees to generate. The question is, can the company control the copy? My working assumption is that this is beyond the scope of any storage system. DRM systems are going to have to come into play and then it's key management on top of that.

MARGO SELTZER: I'm not sure I buy this. Yes, companies care that employees do their jobs, but very few companies tell their employees how to do their job. If my job is to produce some information and data, I may be traveling for a week and it may take some time for that to happen. In the meantime, I may be producing valuable corporate data on my laptop that is not yet on any corporate server. Whether it gets there or not is a process issue and process issues don't always get resolved in the way we intend.

MACHE CREEGER: You're both right. Margo wants to create value for her company in whatever way she is comfortable—on a laptop while she's traveling, at home—whichever way works that produces the highest value for her employment contract. If the company values Margo's work, they will be willing to live, within reason, with Margo's work style.

On the other hand, from Steve's perspective, sooner or later, Margo will have to take what is a free-form edge document and check it into a central protected repository and live with controls. She can then go on to the next production phase, which might be a Rev. 2 derivative of that original work, or perhaps something completely different.

ERIK RIEDEL: You certainly have to be careful. You're moving against the trend here. The trend is toward decentralization. Corporations are encouraging people to work on the beach and at home.



ERIK RIEDEL

The general trend for the last several years is for more distribution, not less. People use a lot of high-capacity, portable devices of all sorts, such as BlackBerrys, portable USB devices, and laptops. For a system administrator, the ability to capture data is much more threatening today.



STEVE KLEIMAN: Nothing I've said is in conflict with that. Essentially, the distilled intellectual property has to come back to the corporation at some point.

MARGO SELTZER: Sometimes it's the process that's absolutely critical. Did I steal the code or write it myself? That information is only encapsulated on my laptop. Regardless of whether I check it into Steve's repository, when Mary's company comes and sues me because I stole her software, what you really care about is the creation process that did or did not happen on my laptop.

ERIC BREWER: I don't think that's the day-to-day problem of a storage administrator. What we're talking about is whether the first goal is to know which of the copies you don't want to lose, which is a different problem than copies leaking out to others.

STEVE KLEIMAN: I do think that the legal system still counts. Technology can't make that obsolete. You still have a legal obligation to a company. You still have an obligation not to break the law. Any technology that we can come up with, someone will probably find a way of circumventing it, and that will require the legal system to fill in the gaps. That's absolutely true with all the stuff on laptops that we don't know how to control right now.

MARGO SELTZER: I also think it's more than just copies that we need to be concerned with; it's also derivative works, to use the copyright term. It's "Oh, look: File A was an input to File B which was an input to File C and now I have File D, and that might actually be tainted because I can see the full path of how it got there."

MACHE CREEGER: Maybe what we're seeing here is that we need to intuit more semantics about the bits we are storing. Files are not just a bunch of bits; they have a history and fit in a context, and to solve these kinds of problems, companies are going to have to put processes and procedures in place to define the context of the storage objects they want to retain.

MARY BAKER: You can clamp down to some extent, but it's the hidden channel problem, even through processes that are not malicious. Say I'm on the beach and the only thing I've got is a non-company PDA and I have some ideas or I talk to somebody and I record something. It can be very hard to

bring all these different sources into a comprehensive storage management policy. Storage has gotten so cheap; it's in everything around us. It's very easy to store bits in lots of places that may be hard to incorporate as part of an integrated system.

STEVE KLEIMAN: There's not just one answer to these problems. Look at what happens in the virus scanning world. It's very much a belt and suspenders approach. They do it on laptops, on storage system, in networks, and on gateways. It's a hard problem, no doubt about it.

There are a variety of technologies for outsourcing markets, such as China and India, where people who are working on a particular piece of source code for a particular company are restricted from copying that source code in any way, shape, or form. The software disables that.

Similar things are possible for the information proliferation issues we have been talking about. All these types of solutions have pros and cons and depend on what cost you are willing to pay. This is not just a technological issue or a storage issue; it's a policy issue that also includes management and legal issues.

ERIC BREWER: In some ways it's a triumph of the storage industry that we have moved from the main concern being how to store stuff to trying to manage the semantics of what we're storing.

MACHE CREEGER: Again, from the storage manager's standpoint, what is he to do? What should he be doing in the next 18 to 24 months?

STEVE KLEIMAN: Today people are saving a lot of time, money, and energy doing server virtualization and storage virtualization. Those two combined are very powerful and I think that's the next two, three, or four years right there.

GREG GANGER: And the products are available now. Multiple people over the course of time have talked about snapshots. If you're running a decent-sized IT operation, you should make sure that your servers have the capability of doing snapshots.

ERIC BREWER: On the security side, encryption. Sometimes there are limited areas where you can do the right kind of key management and hierarchies, but encryption is an established way

in the storage realm to begin to protect the data in a comprehensive way.

MARGO SELTZER: Backup, archival, and disaster recovery are all vital functions, but they're different functions and you should actually think carefully about what you're doing and make sure that you're doing all three.

GREG GANGER: Your choice for what you're doing for any one of the three might be to do nothing, but it should be an explicit choice, not an implicit one.

ERIK RIEDEL: And the other way around. When we're talking about energy efficiency, being efficient about copies, and not allowing things to leak, then you want to think explicitly about why you are making another copy.

ERIC BREWER: And which copies do you really not want to lose? I differentiate between master copies, which are the ones that are going to survive, and cache copies, which are ones that are intentionally transient.

GREG GANGER: For example, if you're running an organization that does software development, the repository, CVS, SVN—whatever it is that you're using—is much more important than the individual copies checked out to each of the developers.

ERIC BREWER: It's the master copy. You've got to treat it differently. No one can weaken your master copy.

MACHE CREEGER: I know that the first CAD systems were developed for and by computer people. They did them for IC chip and printed circuit board design and then branched out to lots of other application areas.

Is the CVS main development tree approach going to be applicable to lots of different businesses and areas for storage problems or do you think the paradigm will be substantially different?

GREG GANGER: It will absolutely be relevant to lots of areas.

ERIC BREWER: I think most systems have cache copies and master copies.

GREG GANGER: In fact, all of these portable devices are fundamentally instances of taking cached copies of stuff.

ERIC BREWER: Any device you could lose ought to contain only cache copies.

MARGO SELTZER: Right, but the reality of the situation is that there are a lot of

portable devices you can lose that are the real copy. We've all known people who've lost their cell phone and with it, every bit of contact information in their lives.

GREG GANGER: They have learned an important lesson and it never happens to them again.

MARGO SELTZER: No, they do it over and over again, because then they send mail out to their Facebook networks that says "send me your contact information."

MACHE CREEGER: They rebuild from the periphery.


ERIC BREWER: The periphery is the master copy; that's exactly right.

MACHE CREEGER: We've talked about security and storage infrastructure. We've touched on copyright, archival, and talked a lot about energy. We've talked about various architectures and argued passionately back and forth between repositories and the free cloud spirit.

Storage managers have a huge challenge. They don't have the luxury of taking the long view of seeing all these tectonic forces moving. They have to make a stand today. They've got a fire hose of information coming at them and they have to somehow structure it to justify their job. They have to do all of this, with no thanks or gratitude from management, because storage is supposedly a utility. Like the lights and the plumbing, it should just work.

STEVE KLEIMAN: They have a political problem as well. The SAN group will not talk to the networking group. The backup group is scared that their jobs are going to go away. Looking at the convergence of technologies, even for something simple like FCoE (Fibre Channel over Ethernet), the SAN Fibre Channel people are circling the wagons.

MACHE CREEGER: Or iSCSI over 10 gigabit Ethernet.

STEVE KLEIMAN: Absolutely. There are a lot of technical issues in there, but there are very serious people and political issues as well. 

Mache Creeger (mache@creeger.com) is a longtime technology industry veteran based in Silicon Valley. Along with being a columnist for *ACM Queue*, he is the principal of Emergent Technology Associates, marketing and business development consultants to technology companies worldwide.

introducing...

ACM's *Newly Expanded* Online Books & Courses Programs!

Helping Members Meet Today's Career Challenges

3,000 Online Courses from SkillSoft

The ACM Online Course Collection features **unlimited access to 3,000 online courses** from SkillSoft, a leading provider of e-learning solutions. This new collection of courses offers a host of valuable resources that will help to maximize your learning experience. Available on a wide range of information technology and business subjects, these courses are open to ACM Professional and Student Members.



SkillSoft courses offer a number of valuable features, including:

- **Job Aids**, tools and forms that complement and support course content
- **Skillbriefs**, condensed summaries of the instructional content of a course topic
- **Mentoring** via email, online chats, threaded discussions - 24/7
- **Exercises**, offering a thorough interactive practice session appropriate to the learning points covered previously in the course
- **Downloadable content** for easy and convenient access
- **Downloadable Certificate of Completion**

"The course Certificate of Completion is great to attach to job applications!"

ACM Professional Member

600 Online Books from Safari

The ACM Online Books Collection includes **unlimited access to 600 online books** from Safari® Books Online, featuring leading publishers including O'Reilly. Safari puts a complete IT and business e-reference library right on your desktop. Available to ACM Professional Members, Safari will help you zero in on exactly the information you need, right when you need it.



Association for
Computing Machinery

Advancing Computing as a Science & Profession

500 Online Books from Books24x7

All Professional and Student Members also have **unlimited access to 500 online books** from Books24x7®, in ACM's rotating collection of complete unabridged books on the hottest computing topics. This virtual library puts information at your fingertips. Search, bookmark, or read cover-to-cover. Your bookshelf allows for quick retrieval and bookmarks let you easily return to specific places in a book.



pd.acm.org
www.acm.org/join

The answer to software reliability concerns may lie in formal methods.

BY MIKE HINCHEY, MICHAEL JACKSON, PATRICK COUSOT, BYRON COOK, JONATHAN P. BOWEN, AND TIZIANA MARGARIA

Software Engineering and Formal Methods

THE SOFTWARE ENGINEERING community has devised many techniques, tools, and approaches aimed at improving software reliability and dependability. These have had varying degrees of success, some with better results in particular domains than others, or in particular classes of applications. A popular, although not uncontroversial, approach is known as *formal methods*, whereby a specification notation with formal semantics, along with a deductive apparatus for reasoning, is used to specify, design, analyze, and ultimately implement a hardware or software (or hybrid) system.

Such an approach is often thought to be difficult to apply and to require significant mathematical experience.^{1,11} Experience has demonstrated that developers without significant mathematical ability can at least understand and use formal specifications,

even if greater mathematical ability and *specialization* in various areas of mathematics and engineering are needed for more challenging formal methods-related activities such as verification and refinement, and even the task of writing formal specifications themselves. As automated formal methods slowly become a natural part of the design process, we also experience higher dependability levels as a standard trait of software products.

A key issue, however, is the need for those applying formal methods to be able to abstract and to model systems at an appropriate level of representation—that is, to develop solid design principles and apply them to software development. This is particularly true when proving properties of more complex systems involving significant concurrency and interaction among components. When we wish to prove properties, it is often easier—and necessary—to prove them at a more abstract level, exploiting the idea of *abstract interpretation*.

This article contains contributions from three world-renowned experts in the fields of software engineering, abstract interpretation, and verification of concurrent systems: Michael A. Jackson, Patrick Cousot, and Byron Cook. Their contributions are based on their keynote speeches at the IEEE's Fifth International Conference on Software Engineering and Formal Methods, held in London, Sept. 10–14, 2007. The aim of the conference series is to bring together practitioners and researchers in the fields of formal methods and software engineering with the goal of exploiting synergies and furthering our understanding of specialization, abstraction, and verification techniques, among other areas.

Declining Dependability Levels

Computer-based systems are pervasive and now influence almost every facet of our lives. They wake us in the morning, control the cooking of our food, entertain us in the guise of media players, help in avoiding traffic congestion, control or identify (via GPS navigation

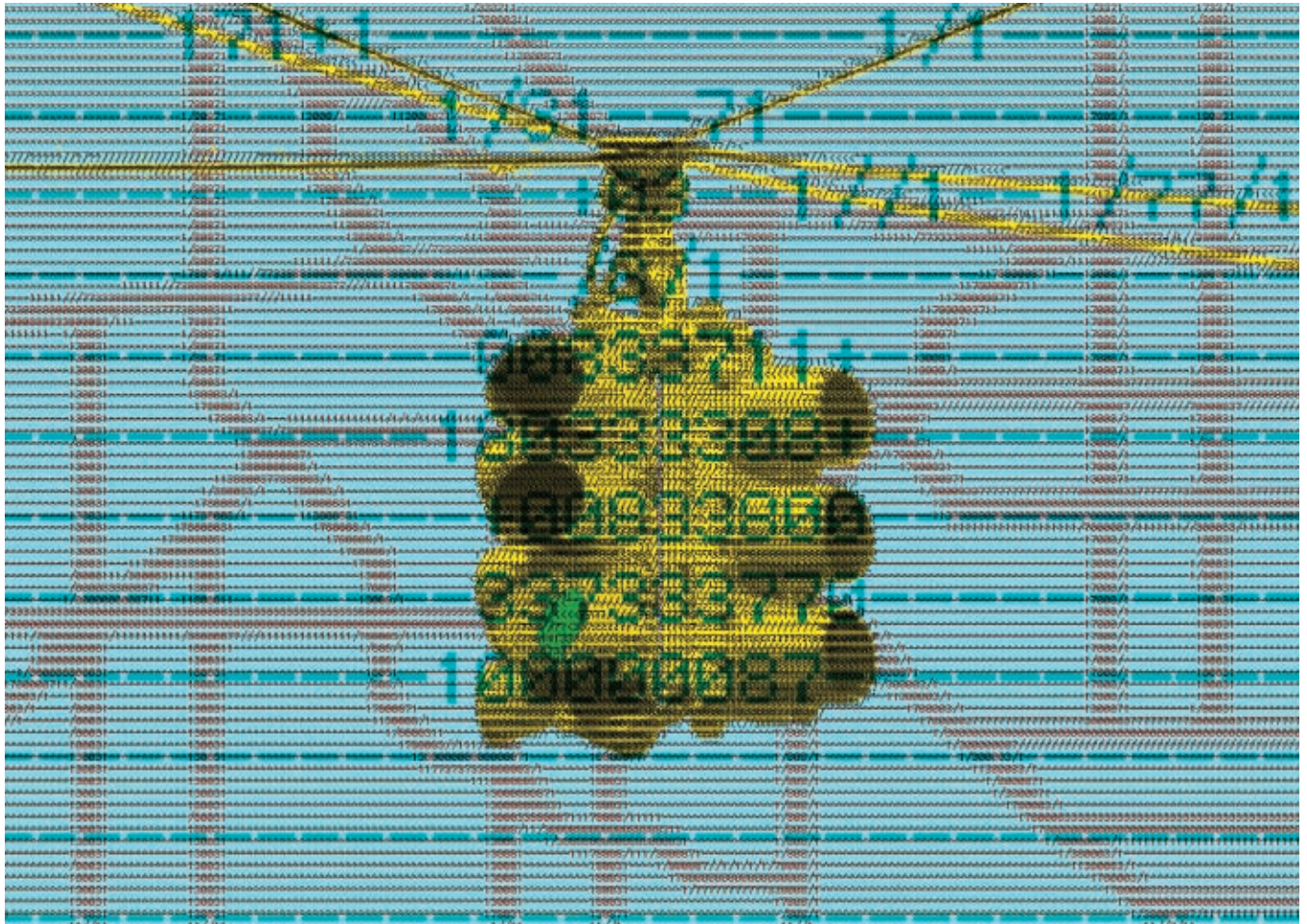


ILLUSTRATION BY JOE MAGEE

systems) the location of the vehicles in which we travel, wash our clothes, and even give us cash from our bank accounts (sometimes!).

Computers are being used increasingly in applications where they can have great influence over our very existence.² They control the flow of trains in the subway, signaling on railway lines, even traffic lights on the street. The failure of any of these systems could cause great inconvenience and conceivably result in accidents in which lives are lost. As they control the actual flight of an aircraft, cooling systems in chemical plants, feedback loops in nuclear power stations, and so on, we can see that they all account for the possibility of great disasters if they fail to operate as expected.

More and more, these systems are *software intensive*, meaning that software is the major component and that much of the functionality is achieved via software rather than hardware implementations. This raises questions over the *reliability* (the measure of the ability of system to continue operating

over time¹²) and the *dependability* (the property that reliance can justifiably be placed on the service it delivers¹²) of software.

A Software Crisis

This has become a major issue for the software engineering community. While hardware dependability has increased continually over the years, and with mean time to failure (a measure of dependability) for the most reliable systems now exceeding 100 years,¹³ software has not kept up with this pattern and indeed has been exhibiting declining levels of dependability.¹⁰

This can be attributed to many factors, including:

- ▶ A naïve belief that anyone who can write software can write *good* software.
- ▶ A mistaken belief that running a few representative test cases indicates that the software is “correct” or adequate.
- ▶ Failure to understand that realizing a good *design* is more important than producing vast quantities of code and that the goal of software engineering is not only to produce code but also to pro-

duce trustworthy solutions to problems that can eventually be implemented in a programming language.

▶ Failure to realize that making changes to software—and in particular unnecessary, uncontrolled, and careless changes—can have an effect on its appropriateness and validity, its correct operation, and can make it less efficient, or in extreme cases obsolete.

Of course, software is *intended* to change, and must be able to change. If we were to write software that we would never change after deployment (to meet changing requirements, an evolving environment, or to correct errors or unimplemented or incorrectly implemented requirements), then we would be better off implementing everything in hardware; but this is neither technically possible nor financially or spatially feasible.

Michael Jackson: Specializing in Software Engineering

Software-intensive systems are intended to interact dependably with the human and physical problem world. Execution

of the software produces and uses real-time information about the world, and monitors and partly controls its behavior to provide required functionality and to satisfy required constraints.

Such systems pose a particular challenge, arising from the interaction between the quasi-formal nature of the software and the nonformal nature of its human and physical problem world outside the computer. The software in execution can be regarded as a formal system: for all but the most extremely critical systems the computer behavior can be assumed to conform to the program semantics. The nonformal problem world, in contrast, has many parts—human, natural, and engineered—whose properties and behavior are far less reliable. For a dependable system, there must be an adequate formal model of the problem world, and the software must be designed to reflect the assumption that the world conforms to this model. In executing its program, the computer always behaves as if the world model is valid; if the world deviates from the model, the system will fail in some way. The problem world, being nonformal, has unbounded possible behaviors and properties that can invalidate any formal model. Fault-tolerant techniques in which the model is weakened in suitably chosen respects can mitigate the difficulty, but they cannot eliminate it: the world can still invalidate the weakened model. How, then, can a dependable system be designed?

Development of the formal model is, above all, an engineering task. In the established engineering branches, the challenge is met by experience accumulated in each particular product class and captured in a *normal design discipline*. The aeronautical engineer W. G. Vincenti explains normal design: “The engineer knows at the outset how the device in question works, what are its customary features, and that, if properly designed along such lines, it has a good likelihood of accomplishing the desired task.” Family cars, for example, are the product of an industry-wide normal design discipline. Different manufacturers’ products are strikingly similar both in their external appearance and in their internal workings because their designers adhere closely to the current normal design. This normal design embodies the accumulated knowledge of the

models—both of the product itself and of its environment or problem world—that is adequate for a desired level of dependability: which deviations from a model are sufficiently improbable to be ignored; which are implicitly handled by the normal design configuration; and which must be explicitly considered in the calculations and checks mandated by the discipline of the normal design practice.

The engineer lacking an applicable normal design discipline must inevitably engage in *radical design*. In Vincenti’s words: “In radical design, how the device should be arranged or even how it works is largely unknown. The designer has never seen such a device before and has no presumption of success. The problem is to design something that will function well enough to warrant further development.” In short, radical design has a low expectation of dependability: it can be a good choice only in purely experimental work or where novelty and excitement are of the highest importance and dependability is little valued.

A normal design discipline does not arise easily or by chance. It rests on a culture of *specialization*, in which a community of organizations and individuals is devoted to gradual, long-term evolution of normal design in a particular product class and of the scientific and engineering knowledge it embodies. The prerequisites for this culture of specialization are two: the continuing existence of specialized communities, with their research and production facilities and their journals and conferences; and the desire of individual highly talented engineers in each generation to dedicate their professional careers to working within just one specialization. This infrastructure of a specialized community and the advances needed to evolve a successful normal design discipline are not achievable or sustainable by floating populations of itinerant generalists.

Software engineering and computer science already have many specializations: some in complexity theory, concurrency, real-time systems, programming languages, model checking, program proving, and distributed computing; and some in system software components such as compilers, database systems, virus checkers, and SAT (Boolean satisfiability problem)

solvers. These specializations, however, are not enough. To build dependable software-intensive systems, we must emulate the established engineering branches, developing many more specializations that are sharply focused on narrow classes of end products and their component subsystems. Only the product-oriented specialist can achieve dependability by bringing together the contributions of specialists in all the different aspects, parts, and dimensions of the design task.

What particular specializations, then, are needed? We cannot answer this question in advance, or in any systematic or authoritative way. In a society that insists on dependable systems, with a software-engineering culture that values and motivates specialization, particular specializations will emerge and evolve in response to opportunities and challenges as they are recognized.

In sum, dependability in an engineering product must be based on a normal design discipline. Only in a product-oriented normal design discipline can all the necessary knowledge—tacit and explicit—be brought together to build a dependable system. This normal discipline in turn must be the product of a long-term community of engineers specializing in systems of the particular product class. For a critical software-intensive system, specialization is not optional.

Patrick Cousot: The Role of Abstract Interpretation in Formal Methods

Formal Verification Methods. In computer science and software engineering, *formal methods* are mathematically based techniques for the specification, development, and verification of software and hardware systems. They establish the satisfaction of a required property (called the *specification*) by a formal model (called the *semantics*) of the behavior of a system (for example, a program and its physical environment). The *semantic domain* is a set of all such formal models of system behaviors.

A *property* of the system is a set of semantic models that satisfy this property. The *satisfaction* of a specification by a system (more precisely by its semantics), which can equivalently be defined as the proof that its strongest property is a property of the system, is called its *collecting semantics*.

An example is the *trace semantics* of a programming language. The semantics of programs in the language describes all possible program executions as a set of traces over states chosen in a given set of possible states. Two successive states in a trace correspond to an elementary program computation step. An example of a property is the termination property stating that all execution traces should be finite.

Formal verification methods are very hard to put in practice because both the semantics and the specification of a complex system are extremely difficult to define. Even when this is possible, the proof cannot be automated (using a theorem prover, a model checker, or a static analyzer) without great computational costs. Considering all possible systems is even harder. Hence it is necessary either to work in the small (for example, model checking) or to consider approximations of large systems (for example, abstract model checking).

Abstract Interpretation. Abstract interpretation⁵ is a theory of sound approximation of mathematical structures, in particular those involved in the description of the behavior of computer systems. To prove a property, the *abstraction* idea is to consider a sound overapproximation of the collecting semantics, a sound underapproximation of the property to be proven, and to make the correctness proof in the abstract.

For automated proofs, these abstractions must be computer-representable, so they are not chosen in the mathematical concrete domain but in an *abstract domain*. The correspondence is given by a *concretization function* mapping abstract properties to corresponding concrete properties. Formal verification being undecidable, the abstraction may be *incomplete*—that is, it produces *false alarms*, a case when a concrete property holds but this cannot be proved in the abstract for the given abstraction, which must therefore be refined. The *abstraction function* is the inverse of the concretization function. It maps concrete properties to their approximation in the abstract domain. Applied to the collecting semantics of a computer system, it formally provides an abstraction of the properties of the system. Abstract verification methods consist of designing an abstraction function that is coarse



MICHAEL JACKSON

Development of the formal model is, above all, an engineering task. In the established engineering branches, the challenge is met by experience accumulated in each particular product class and captured in a normal design discipline.



enough so that the abstract collecting semantics is computer-representable and effectively computable and is precise enough so that the abstract-collecting semantics implies the specification.

Abstract interpretation formalizes the intuition about abstraction. It allows the systematic derivation of sound *reasoning methods* and *effective algorithms* for approximating undecidable or highly complex problems in various areas of computer science (semantics, verification and proof, model checking, static analysis, program transformation and optimization, typing, or software steganography). Its main current application is on the safety and security of complex hardware and software computer systems.

Verification by Static Analysis. *Static code analysis* is the fully automatic analysis of a computer system by direct inspection of the source or object code describing the system with respect to the semantics of the code (without executing programs, as in *dynamic analysis*). The proof is done in the abstract by effectively computing an abstraction of the collecting semantics of the system. Examples of successful static analyzers used in an industrial context are aiT (www.absint.com/ait used to compute an overapproximation of the worst-case execution time⁸) and Astrée (www.astree.ens.fr used to compute an overapproximation of the collecting semantics to prove the absence of runtime errors⁶).

Their growing success is a result of their being useful (such as, they tackle practical problems), sound (their results can be trusted), nonintrusive (end users do not have to alter their programming methods (for example, by producing the specification and abstract semantics that can be directly derived from the program text), realistic (applicable in any weird industrial environment), scalable (to millions of lines of code as found in actual industrial code), and conclusive (producing few or no false alarms).

The design of language-based semantics and abstractions is extremely difficult but possible on a well-defined family of programs (for example, synchronous, time-triggered, real-time, safety-critical, embedded software written or automatically generated in C for

Astrée). The abstraction of Astrée is designed by conjunction of many elementary abstractions that are individually simple to understand and implement. In the case of a false alarm, each abstraction can be adjusted in cost and precision by parameters and abstraction directives (whose inclusion in the code can be automated). If the false alarm cannot be solved by the existing abstractions, new abstractions can be easily incorporated to extend and refine the conjunction of abstractions. The abstract invariants can therefore be strengthened in a few refinement steps until no false alarm is left.⁷

Byron Cook:

Verification of Concurrent Systems

Many critical applications involve high degrees of concurrency, whereby a number of activities progress in parallel with each other. Concurrency can create surprisingly complex interactions among the concurrent components, making the verification problem inherently more complex.

The traditional methods of specifying and automatically reasoning about computer systems are not sufficient for use with concurrent systems. They do not allow for the side effects caused by other concurrent components, the occurrence of multiple simultaneous events, or the synchronization required between processes to ensure data integrity, and so forth. Specialized tools will normally be required for effective verification.

As an example, consider thread termination. Concurrent programs are often designed such that certain functions executing within critical threads must terminate. Such functions can be found in operating systems, Web servers, and email clients. Until now, no known automatic program termination prover supported a practical method of proving the termination of threads.

As another example, concurrent programs are usually written in languages that assume but do not guarantee memory safety (such as, pointer dereferences never fail, and memory is never leaked). Again, until now, no known automatic program verifier supported the proving of memory safety for concurrent programs.

Recent advances now allow us to address these problems (such as, proving



BYRON COOK

Concurrency can create surprisingly complex interactions among the concurrent components, making the verification problem inherently more complex.



properties such as memory safety and termination of concurrent code. These advances have led to the recently added support for concurrency in the Terminator termination prover¹⁵ and the SLayer shape analysis engine.¹⁴

The common theme of these advances is thread modularity: existing sequential-program provers can be used to prove the correctness of concurrent programs if appropriate abstractions can be found to represent the other threads in a concurrent system. In the case of termination proving, the abstractions describe the direction of change (called the variance) of values caused by the other threads in the system. We might know, for example, that the value of the variable x cannot go up in the other threads, thus allowing us to prove the termination of a loop in the thread of interest that decrements x during each loop iteration. In the case of memory-safety proving, the abstractions describe the association between locks and data structures: we can prove the memory safety of the thread of interest by assuming that certain data structures are safe to modify only when the associated locks are acquired.

The difficulty with thread-modular techniques is that the space of abstractions is so large that finding the right one for a given program verification problem is difficult. Heuristics must be developed. Furthermore, through the use of experimental evaluations, these heuristics must be shown to work in the common case. We have developed and evaluated such heuristics. In the case of termination proving, we have found that by temporarily ignoring concurrency we can guess a set of ranking functions (see Patterson¹³ for details), which leads to a useful candidate abstraction. Using static program analysis techniques we can prove, for example, that every instruction in the other threads does not increment x . The fact that x needs not to be incremented, as opposed to y , comes from the proof of termination via the sequential-program termination prover (see Cook⁴ for the details).

In the case of proving memory safety, we use initial guesses that match locks to variables, and then use counterexample-guided techniques to refine the shape of the data structures protected by the locks (see Gotman⁹ for further details).

Conclusion

Dependability in software-intensive systems can be achieved only through the application of solid design principles. That, in turn, is achieved through an understanding of the product and specialization of engineers in particular domains, product types, and techniques. Abstract interpretation can aid in reducing the complexity inherent in proving properties and correctness of complex software systems, which otherwise would not be feasible. Such an approach facilitates automated reasoning and fully automated reasoning. In particular, recent advances in verification techniques have made it possible to verify concurrent systems and prove termination by considering termination of a thread without considering concurrency per se.

In the realm of formal methods, too, we encounter radical and normal engineering, as well as the whole spectrum of intermediate phases in between. In a sense, the ratio of radical design to normal design can be taken as a measure of the maturity of a field of engineering.

In retrospect, radical engineering of, and with, formal methods has characterized the first 20 years of this field, making it an art: it was limited to a restricted scope of few users and a few high-budget, high-risk projects for which high assurance was compelling. This is no longer the case today:

► A first transition from weak to strong formal methods¹⁷ moved the field from specification-only toward tool-based semantics analysis, making formal methods not only descriptive but also operational.


► A second transition shifted tool support from heavyweight to lightweight formal methods: from proof assistants, which still require specialist skills, education, and a good deal of intuition, to fully automatic analyses embedded in a software engineer's usual development environment.

This shift is central to enabling every software developer to use the techniques seamlessly, advancing from a radical design endeavor to everyday practice. Then it is not an art anymore, just another craft, in the same way that photography enabled everybody to portray a subject with perfect faithfulness to the subject's traits, at a very reasonable cost and without being, or resort-

ing to, a portrait painter.

We observe that today different techniques occupy different positions on this transition axis. While termination proof of concurrent systems is still an art (a case for radical design), techniques such as type checking are enforced by the vast majority of programming environments (natural design), and many other techniques are walking the naturalization path. It is a steep and narrow path that takes decades to follow. For example, abstract interpretation was discovered in the 1970s, but it is only in recent years that we have had tools such as Astrée to enable normal developers to apply collections of sophisticated abstract interpretations as part of their daily routine within an enhanced normal engineering process. Model checking, rewarded this year with the ACM A.M. Turing Award to Edmund M. Clarke, Joseph Sifakis, and E. Allen Emerson, is another successful example of formal methods that became increasingly natural in the course of their 25+ years of history. Once naturalized, the magic of a technique (and of its gurus) vanishes, but we all profit from the achievements of the revolutionaries that enter mainstream production.

As noted 12 years ago, specialization in high-assurance systems concerns devising appropriate heterogeneous methods that adequately exploit the various application-specific characteristics of the problem.¹⁶ Computer-aided *method engineering* is the new craft. It targets understanding and solving problems *heterogeneously* at a meta level, where whole methods and paradigms are combined. Even though this holds already for many sequential systems, it is particularly true for distributed systems, which by their nature are of a much higher conceptual complexity. Multicore architectures in our laptops and massively multicore systems as part of Web computing and cloud computing environments demand increased attention here.

This quest continues... 

References

1. Bowen, J.P. and Hinchey, M.G. Seven more myths of formal methods. *IEEE Software* 12, 4 (July 1995), 34–41.
2. Bowen, J.P. and Hinchey, M.G. High-integrity system specification and design. *Formal Approaches to Computing and Information Technology*. Springer-Verlag, London, 1999.
3. Cook, B., Podelski, A., Rybalchenko, A. Termination proofs for systems code. In *Proceedings of the*

2006 ACM SIGPLAN Conference on Programming Language Design and Implementation, 415–426.

4. Cook, B., Podelski, A., Rybalchenko, A. Proving thread termination. In *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2007*, 320–330.
5. Cousot, P. and Cousot, R. Systematic design of program analysis frameworks. In *Conf. Rec. 6th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Prog. Lang.*, ACM Press (1979), pp 269–282.
6. Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., and Rival, X. Varieties of static analyzers: A comparison with Astrée. In *Proceedings of the 1st IEEE & IFIP Int. Symp. on Theoretical Aspects of Software Engineering, TASE '07*. IEEE Computer Society Press (2007), 3–17.
7. Delmas, D. and Souyris, J. Astrée: From research to industry. *Lecture Notes in Computer Science* 4634, Springer (2007), 437–451.
8. Ferdinand, C., Heckmann, R., and Wilhelm, R. Analyzing the worst-case execution time by abstract interpretation of executable code. *Lecture Notes in Computer Science* 4147, Springer (2006), 1–14.
9. Gotsman, A., Berdine, J., Cook, B., Sagiv, M. Thread-modular shape analysis. In *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation*, 266–277.
10. Gray, J. Dependability in the Internet era. In *Proceedings of the High Dependability Computing Consortium Conference*, Santa Cruz, CA, May 7, 2001.
11. Hall, J.A. Seven myths of formal methods. *IEEE Software* 7, 5 (Sept. 1990), 11–19.
12. Laprie, J.-C., ed. Dependability: Basic concepts and terminology in English, French, German, Italian and Japanese. *Dependable Computing and Fault-Tolerant Systems*, Vol. 5. Springer-Verlag, NY, 1992.
13. Patterson, D. et al. *Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies*. Computer Science Technical Report UCB//CSD-02-1175. University of California, Berkeley, CA, March 15, 2002.
14. <http://research.microsoft.com/SLayer>
15. <http://research.microsoft.com/terminator>
16. Steffen, B., Margaria, T. Method engineering for real-life concurrent systems. *ACM Computing Surveys, Special Issue: Position Statements on Strategic Directions in Computing Research* 28, 4es (Dec. 1996) 56. ACM, NY.
17. Wolper, P. The meaning of "formal": From weak to strong formal methods. *STTT* 1, 1–2 (1997), 6–8. Springer Verlag.

Mike Hinchey (mike.hinchey@lero.ie) is codirector of Lero, the Irish Software Engineering Research Center, and professor of software engineering at the University of Limerick, Ireland.

Michael Jackson (jacksonma@acm.org) is Visiting Research Professor, Centre for Research in Computing, The Open University, Milton Keynes, England.

Patrick Cousot (Patrick.Cousot@ens.fr) is a professor of computer science at the École Normale Supérieure. He is a specialist in semantics, verification, and static analysis of programs and complex systems and is the inventor of abstract interpretation.

Byron Cook (bycook@microsoft.com) is a researcher at Microsoft's Laboratory at Cambridge University, where he has been working on the program termination prover Terminator, the shape analysis engine SLayer, and the software model checker SLAM.

Jonathan P. Bowen (jpbowen@gmail.com) is chairman of Museophile Limited. He is contracted to work for Praxis High Integrity Systems, applying formal methods for software testing. He is also a visiting professor at King's College London and an emeritus professor at London South Bank University.

Tiziana Margaria (margarita@cs.uni-potsdam.de) is chair of service and software engineering at the Institute of Informatics, Universität Potsdam, Germany. She is also president of the European Association of Software Science and Technology (EASST).

DOI:10.1145/1378727.1378743

Beyond Google, emerging question-answering systems respond to natural-language queries.

BY DMITRI ROUSSINOV, WEIGUO FAN, AND JOSÉ ROBLES-FLORES

Beyond Keywords: Automated Question Answering on the Web

SINCE THE TYPICAL COMPUTER USER spends half an hour a day searching the Web through Google and other search portals, it is not surprising that Google and other sellers of online advertising have surpassed the revenue of their non-online competitors, including radio and TV networks. The success of Google stock, as well as the stock of other search-portal companies, has prompted investors and IT practitioners alike to want to know what's next in the search world.

The July 2005 acquisition of AskJeeves (now known as Ask.com) by InterActiveCorp for a surprisingly high price of \$2.3 billion may point to some possible

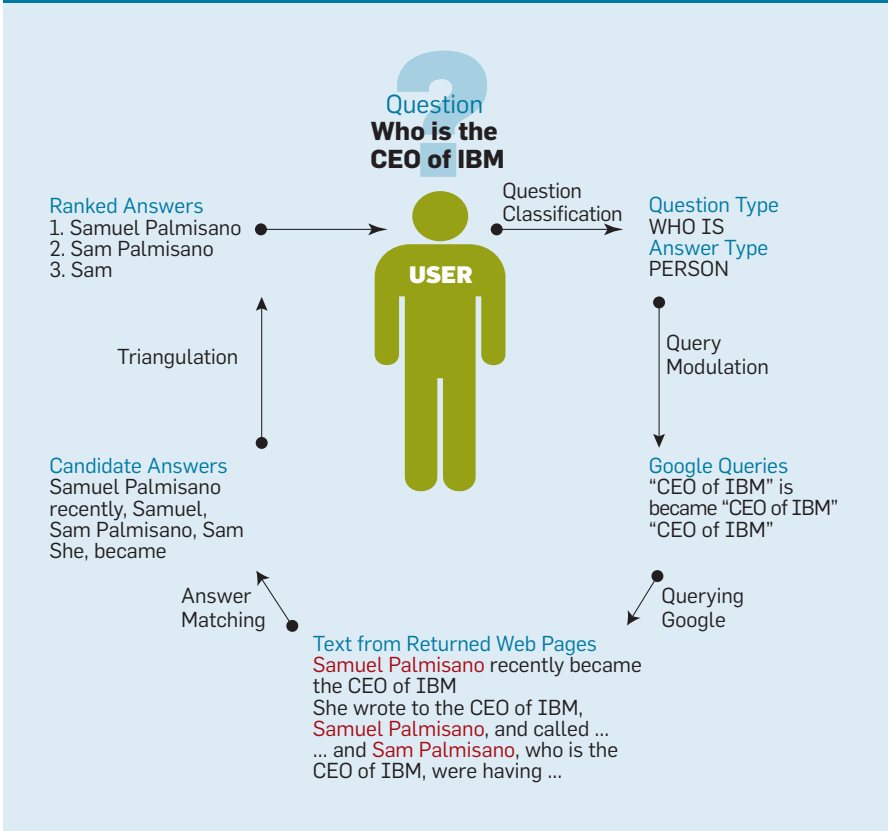
answers. Ask.com not only wanted a share of the online-search market, it also wanted the market's most prized possession: completely automated open-domain question answering (QA) on the Web, the holy grail of information access. The QA goal is to locate, extract, and provide specific answers to user questions expressed in natural language. A QA system takes input (such as "How many Kurds live in Turkey?") and provides output (such as "About 15 million Kurds live in Turkey," or simply "15 million").

Search engines have significantly improved their ability to find the most popular and lexically related pages to a given query by performing link analysis and counting the number of query words. However, search engines are not designed to deal with natural-language questions, treating most of them as "bags," or unordered sets, of words. When a user types a question (such as "Who is the largest producer of software?"), Google treats it as if the user typed "software producer largest," leading to unexpected and often not-useful results. It displays pages about the largest producers of dairy products, trucks, and "catholic software," but not the answer the user might expect or need (such as "Microsoft"). Even if the correct answer is among the search results, it still takes time to sift through all the returned results and locate the most promising answer among them.

It is more natural for people to type a question (such as "Who wrote King Lear?") than to formulate queries using Boolean logic (such as "wrote OR written OR author AND King Lear"). Precise, timely, and factual answers are especially important when dealing with a limited communication channel. A growing number of Internet users have mobile devices with small screens (such as Internet-enabled cell phones). Military, first-responder, and security systems frequently put their users under such time constraints that each additional second spent browsing search results could put human lives at risk. Finally, visually impaired computer us-



Architecture of a typical Web QA system.



constraints. This emphasizes simple, computationally efficient algorithms and implementations (such as simple pattern matching vs. “deep” linguistic analysis).

A typical Web QA system architecture is illustrated by the NSIR system (see the Figure here),⁵ one of the earliest Web QA systems (1999–2005) developed at the University of Michigan, and the more recent Arizona State University Question Answering system (ASU QA).⁶ When given a natural-language question (such as “Who is the largest producer of software?”), the system recognizes a certain grammatical category (such as “what is,” “who is,” and “where was”), as well as the semantic category of the expected answer (“organization” in this example). NSIR uses machine-learning techniques and a trainable classifier to look for specific words in the questions (such as “when” and “where”), as well as parts of speech (POS) of the other words supplied by the well-known Brill’s POS tagger.² For example, in the question “What ocean did Titanic sink in?” the tagger identifies “ocean” as a noun and “sink” as a verb. The trained classifier classifies the expected answer type as “location.”

ASU QA matches the question to one of the trained regular expressions. For example, the question “What ocean did Titanic sink in?” matches “What <C> did <T> <V>,” where <C> is any word that becomes the expected semantic category (“ocean”), <T> is the word or phrase that becomes the question target (“Titanic”), and <V> is the verb phrase (“sink in”). While NSIR and ASU QA use only a few grammatical and semantic categories, some other (non-Web) systems involve more fine-tuned taxonomies. For example, Falcon,⁷ one of the most successful TREC systems, is based on a pre-built hierarchy of dozens of semantic types of expected answers, subdividing the category “person” further into “musician,” “politician,” “writer,” “athlete,” and more.

Web QA systems generally do not crawl or index the Web themselves. They typically use the “meta engine” approach: send one or more queries to commercial engines providing application programming interfaces (APIs) specifically designed for this purpose. The query-modulation step in the Figure creates requests for the search engine

ers simply cannot enjoy the quantity of information available on the Web, since they are unable to glance through the pages of snippets that are returned by search engines. The best available reader software and refreshable Braille screens do not provide enough bandwidth for real-time interaction.

Although Google and Microsoft have announced that they’ve added QA features to their engines, these capabilities are limited, as we found in the simple experiment we report here and reconfirmed at the time of publication. Since many practitioners are familiar with the concept of online QA, we review only the recent advances in automated open-domain (Web) QA and the challenges faced by QA. We contrast the most noticeable (in terms of academic research interest and media attention) systems available on the Web and compare their performance, as a “team,” against two leading search portals: Google.com and MSN.com.

Technology Foundation

For the past decade, the driving force behind many QA advances has been the annual competition-like Text Retrieval Conference (TREC).⁸ The par-

ticipating systems must identify precise answers to factual questions (such as “who,” “when,” and “where”), list questions (such as “What countries produce avocados?”), and definitions (such as “What is bulimia?”).

The following distinctions separate QA from a fixed corpus (also called “closed domain,” as in TREC competitions) and QA from the entire Web (typically referred to as “open corpus” or open-domain QA):

Existence of simpler variants. The Web typically involves many possible ways for answers to begin, allowing QA fact-seeking systems to look for the lowest-hanging fruit, or most simple statements of facts, making the task easier at times;

Expectation of context. Users of Web-based fact-seeking engines do not necessarily need answers extracted precisely. In fact, we’ve personally observed from our interaction with practitioners (recruited from among our MBA students) that they prefer answers in context to help verify that they are not spurious; and

Speed. Web-based fact-seeking engines must be quick, and TREC competition does not impose real-time

based on the words in the questions that are sometimes expanded with synonyms. For example, ASU QA takes dozens of patterns for each question type from the questions and answers seen previously during the training process. It transforms the question, say, “Who is the CEO of IBM?” into the Google query “became the CEO of IBM” because it has previously seen the answer “Washington” to the question “What is the capital of the U.S.?” in the sentence “Washington became the capital of the U.S. on June 11, 1800.”

Since many of the factual answers are named entities (such as people, organizations, countries, and cities), QA systems typically employ third-party named-entity-identification techniques to extract candidate answers (such as Minipar).¹ All named entities in the proximity of the question words and that match the desired semantic category are identified as candidate answers. Meanwhile, ASU QA, employs a pattern-matching mechanism to perform answer extraction. A sentence like “Samuel Palmisano recently became the CEO of IBM” matches the pattern “<A> became <Q>,” where <A> = “Samuel Palmisano recently” is the candidate answer, and <Q> = “the CEO of IBM” is the question’s focus. ASU QA also treats all subphrases from each candidate answer as candidates themselves. In the example, the subphrases are “Samuel Palmisano recently,” “Samuel Palmisano,” “Palmisano recently,” “Samuel,” “Palmisano,” and “recently.”

In order to identify the most probable (supported) answer, ASU QA has gone several steps beyond frequency counts explored earlier by Dumais et al.¹ and other groups involved in TREC competitions that involved a probabilistic triangulation mechanism. Triangulation is a term widely used in the intelligence and journalism fields for confirming or disconfirming facts by checking multiple sources. Roussinov’s and Robles’s algorithm is demonstrated through the following intuitive example: Imagine that we have two candidate answers for the question “What was the purpose of the Manhattan Project?”: (1) “To develop a nuclear bomb” or (2) “To create an atomic weapon.” They support (triangulate) with each other since they are semantically similar. In the example involving the CEO of IBM, “Samuel

Palmisano” and “Sam Palmisano” win because they reinforce each other.

Although QA technology is maturing quickly and seems promising for a number of practical applications (such as commonsense reasoning and database federation), few QA systems go beyond information seeking. Although the Ford Motor Company and Nike, Inc. began using Ask.com as their site search engine in 2005, they’ve never reported if QA features are indeed practical and useful. In 2005, Roussinov and Robles demonstrated empirically that ASU QA helps locate potentially malevolent online content, potentially helping law-enforcement and public oversight groups combat the proliferation of materials that threaten cybersecurity or promote terrorism.

Feature Comparison

When comparing features and performing our informal evaluation, we chose only the QA systems (see Table 1) mentioned in popular IT magazines or academic publications and that were (and still are) available online during the first run of our study in spring 2005. We did not include Google or MSN since their QA capabilities were (and still are) quite limited. Google occasionally produces precise answers with respect to geography-related questions (such as “What is the population of Cambodia?”) but

does not attempt to target more general or dynamic topics (such as “Who is the CEO of Motorola?”) or more grammatically or semantically challenging questions (“How long can a British Prime Minister serve in office?”). MSN uses only Encyclopedia Encarta as a source of precise answers and is similarly limited in terms of complexity and coverage.

Although AskJeeves enjoyed immense popularity and investor interest at the time it was acquired, its QA capabilities are limited in practice. Its answers to natural-language questions could be provided only from manually created databases, and the topics of inquiry were limited to simple “encyclopedic” requests (such as “What is the population of Uganda?”). When the question does not match any of the anticipated questions, Ask.com would reroute the question as a simple keyword query to its underlying keyword search engine—Teoma, which was acquired by Ask.com in 2001 when it was a failing dot-com based on technology originally created by IBM and further developed at Rutgers University. In 2005, Ask.com introduced certain answer-matching capabilities over the entire Web but is still short of specifying the precise answer while displaying a set of ordered snippets (up to 200) with the words from the highlighted question, similar to Google’s approach.

Table 1: Features of selected Web (open-domain) QA systems.

System	Purpose	Output Format	Multilingual	Technology/ Algorithms	Crawling
AskJeeves	Commercial	Up to 200 rank-ordered snippets	Yes	Undisclosed	Entire Web
BrainBoost	Commercial	Up to 10 snippets or sentences	No	Undisclosed	Meta search
Language Computer Demo	Commercial/ research prototype	Up to 10 snippets	No	Deep parsing, theorem proving, large taxonomy of answer types	Meta search
AnswerBus	Commercial/ research prototype	Up to 10 sentences	Yes	Shallow parsing, entity extraction, small taxonomy of answer types	Meta search
NSIR	Research prototype	Exact answers or snippets	No	Shallow parsing, entity extraction, small taxonomy of answer types	Meta search
ASU QA	Research prototype	Up to 20 snippets	No	Pattern matching, small taxonomy of answer types	Meta search

Since BrainBoost (www.brainboost.com) is a commercial system, little is known outside the company about the algorithms it employs. Nevertheless, it quickly gained popularity among bloggers and other online information seekers, since it delivers decent accuracy and quick response. Answers.com bought BrainBoost for \$4 million

in cash and shares of restricted BrainBoost stock in 2005.

Another prototype developed by Language Computer Corporation (www.languagecomputer.com) returns up to 10 answer snippets, with the words from the question (not the precise answer itself) highlighted. AnswerBus (misshoover.si.umich.edu/~zzheng/qa-new/)

and NSIR (tangra.si.umich.edu/clair/NSIR/html/nsir.cgi) were the two earliest open-domain Web QA systems developed in academic institutions, and their algorithms are detailed in a number of publications.⁵ Based on matching the question to a trained set of answer patterns, ASU QA uses probabilistic triangulation to capitalize on the redundancy of publicly available information on the Web. Along with BrainBoost, ASU QA was used for several years in a \$2 million project supported by NASA (www.aee.odu.edu) aimed at developing collaborative distributed engineering knowledge/information management systems and intelligent synthesis environments for future aerospace and other engineering systems.

Beyond Keywords

Comparing and evaluating different Web QA systems is not straightforward and, to our knowledge, has never been done before the study we describe here. In the annual TREC competition, the rules are set in advance, and participating researchers approximately predict the distribution and types of questions that would be expected from their experience in prior years. Meanwhile, the objectives of each Web QA system are different. The commercial systems (such as Ask.com and BrainBoost) are primarily interested in increasing traffic volume and visibility online to generate maximum potential advertising revenue or investment capital. The research prototypes (such as ASU QA and NSIR) are primarily interested in demonstrating innovative ideas in certain unexplored fields of research involving information seeking, not in competing with commercial systems. As a result, the systems we consider here support different sets of features and interfaces, as in Table 1.

The goal of our study in spring 2005 and repeated in 2007/2008 was not to compare QA systems against each other but to determine whether any of them might offer additional power relative to keyword search engines, exemplified by Google and MSN. In particular, we wanted to know whether automated QA technology provides answers to certain questions that keywords may find difficult or impossible to answer. For this reason, we performed an informal comparison of the QA systems in Table

Table 2: Comparing search-engine performance: Google and MSN (as a team) vs. selected online QA systems (as a team).

Question	Google MRR	MSN MRR	Average MRR for the Search Portals Team	Average MRR for the QA Team
Aspartame is also called what?	0.00	0.33	0.17	0.29
At what speed does the Earth revolve around the sun?	0.00	0.50	0.25	0.25
At what time of year is air travel at a peak?	0.00	0.00	0.00	0.00
Boxing Day is celebrated on what date?	0.50	1.00	0.75	0.63
CNN is owned by whom?	0.20	0.50	0.35	0.65
How big is our galaxy in diameter?	0.14	0.00	0.07	0.65
How did Al Capone die?	0.00	0.00	0.00	0.21
How did Bob Marley die?	0.17	0.00	0.08	0.51
How far is it from Denver to Aspen?	0.11	0.00	0.06	0.29
How far is Pluto from the sun?	1.00	0.11	0.56	0.75
How long can a British Prime Minister serve in office?	0.00	0.00	0.00	0.13
How many copies of an album must be sold for it to be a gold album?	1.00	0.00	0.50	0.35
How many Stradivarius violins were ever made?	0.00	0.00	0.00	0.38
How many teachers are there in the U.S.?	0.00	0.00	0.00	0.08
How much folic acid should an expectant mother get daily?	0.20	0.11	0.16	0.25
In what country is a stuck-out tongue a friendly greeting?	0.25	0.00	0.13	0.00
What color is a giraffe's tongue?	0.50	1.00	0.75	0.63
What continent is Argentina on?	0.33	0.00	0.17	0.63
What continent is Italy on?	0.25	0.00	0.13	0.41
What do you call a professional map drawer?	0.00	0.00	0.00	0.00
What famous model was married to Billy Joel?	1.00	0.13	0.56	0.07
What is the collective noun for geese?	0.17	0.33	0.25	0.75
What is the collective term for geese?	0.33	0.20	0.27	0.81
What is the Islamic counterpart to the Red Cross?	1.00	1.00	1.00	0.68
What is the largest city in Wisconsin?	0.33	1.00	0.67	1.00
What is the largest snake in the world?	1.00	0.50	0.75	0.81
What is the largest variety of cactus?	0.00	0.00	0.00	0.33
What is the most heavily caffeinated soft drink?	0.00	0.00	0.00	0.05
What ocean did the Titanic sink in?	0.20	0.14	0.17	0.68
Average score across all questions:	0.30	0.24	0.27	0.42

1 as a “team” vs. the keyword searching technique (Google and MSN as another “team”). We also wanted to know whether QA might decrease the cognitive load during the answer-seeking process. We therefore claim only that the idea of “going beyond keywords” is possible while searching for answers to questions, not that a particular system is better than another particular system.

No researcher has yet claimed to have produced a representative set of questions for evaluating QA systems. Indeed, such a set might have to include thousands of questions to adequately represent each possible type of question. We built a data set based on our experience with IT practitioners. We merged all the TREC questions with a set of 2,477,283 questions extracted earlier by Radev et al.⁵ from the Excite search engine log of real search sessions.⁵ We then distributed nonoverlapping sets of 100 randomly drawn questions to each of the 16 students in a technology-related MBA class at Arizona State University. The survey was followed by interviews and resulted in the selection of 28 test questions guided by participant choices and comments. In order to avoid researcher bias, it was crucial that we not enter any of the questions into an online system—search engine or QA—before deciding whether to select that particular question for the test.

We used the mean reciprocal rank (MRR) of the first correct answer, a metric also used during the 2001 and 2002 TREC competitions and in several follow-up studies. It assigns a score of 1 to the question if the first answer is correct. If only the second answer is correct, the score is $\frac{1}{2}$. The third correct answer results in a score of $\frac{1}{3}$. The intuition that went into devising this metric is that a reader of online question-answering results typically scans answers sequentially, and “eyeballing” time is approximately proportional to the number of wrong answers before the correct one pops up. However, this computation is known to “misbehave” statistically, being overly sensitive to the cut-off position, the lowest-ranked answer considered,⁵ thus its reciprocals are typically reported and used for averaging and statistical testing. Results are outlined in Table 2.

By rerunning our analysis with each of the members excluded from the QA

team, we verified that no weak players would pull down the QA team’s performance. Because our intention was not to compare individual QA systems, we did not include the data for each individual QA system. The average results support the following observations:

► The QA team performed much better than the keyword-search-engine team, an MRR of 0.42 vs. 0.27; a remarkable 50% improvement was statistically significant, with the p value of the t-test at 0.002;

► The average performance of the QA team is better than the performance of each search engine individually; moreover, each QA system performed better than each keyword search engine;

► For each question to which an answer was found by a keyword search engine, at least one QA system also found an answer; the reverse was not always the case; and

► If a QA system found the correct answer, it was typically second or third in the ranked list; only the fourth or fifth snippet from Google or MSN typically provided the correct answer.

To verify the stability of these observations, we re-ran our tests in spring 2006. Although most of the measurements of the specific systems with respect to the specific questions had changed, their overall performance did not change significantly, and our observations were further reinforced.

Conclusion

Based on our interaction with business IT practitioners and an informal evaluation, we conclude that open-domain QA has emerged as a technology that complements or even rivals keyword-based search engines. It allows information seekers to go beyond keywords to quickly answer their questions. Users with limited communication bandwidth (as a result of small-screen devices or having some visual handicap) will benefit most. And users under some time constraint (such as first responders at a natural disaster) will likely find it more suitable compared to the keywords-to-snippets approach offered by popular search portals like Google and MSN.


However, to compete with established keyword-based search engines, QA systems still must address several technical challenges:

► *Scalability.* Web QA system response

time lags the one-to-two-second performance provided by today’s search engines; more research needs to be done as to how to make Web QA systems more scalable in order to process the comparable loads simultaneously;

► *Credibility.* Information on the Web, though rich, is less factually reliable than counterpart material published on paper; how can QA system developers, as well as search users, factor source credibility into answer ranking?; and

► *Usability.* Designers of online QA interfaces must address whether QA systems should display precise answers, sentences, or snippets.

We look forward to the next five to 10 years for advances in all of them. 

References

- Berwick, R.C. Principles of principle-based parsing. In *Principle-Based Parsing Computation and Psycholinguistics*, R.C. Berwick, S.P. Abney, and C. Tinny, Eds. Kluwer Academic Publishers, Norwell, MA, 1991, 1–38.
- Dumais, S., Banka, M., Brill, E., Lin, J., and Ng, A. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Tampere, Finland, Aug. 11–15). ACM Press, New York 2002, 291–298.
- Kwok, C., Etienne, O., and Weld, D.S. Scaling question answering to the Web. *ACM Transactions on Information Systems* 19, 3 (2001), 242–262.
- Lempert, R.J., Popper, S.W., and Bankes, S.C. *Shaping the Next One Hundred Years: New Methods for Quantitative, Long-Term Policy Analysis*. RAND Corp., Santa Monica, CA, 2003; direct.bl.uk/bld/PlaceOrder.do?UIN=138854587&ETOC=RN&from=searchengine.
- Radev, D., Fan, W., Qi, H., Wu, H., and Grewal, A. Probabilistic question answering on the Web. *Journal of the American Society for Information Science and Technology* 56, 6 (Apr. 2005), 571–583.
- Roussinov, D. and Robles, J. Applying question answering technology to locating malevolent online content. *Decision Support Systems* 43, 4 (Aug. 2005), 1404–1418.
- Surdeanu, M., Moldovan, D.I. and Harabagiu, S.M. Performance analysis of a distributed question/answering system. *IEEE Transactions on Parallel and Distributed Systems* 13, 6 (2002), 579–596.
- Voorhees, E. and Buckland, L.P., Eds. *Proceedings of the 13th Text Retrieval Conference TREC 2004* (Gaithersburg, MD, Nov. 16–19). National Institute of Standards and Technology, Gaithersburg, MD, 2004; trec.nist.gov/pubs/trec13/t13_proceedings.html.

Dmitri Roussinov (Dmitri.Roussinov@cis.strath.ac.uk) is a senior lecturer in the Department of Computer and Information Sciences at the University of Strathclyde, Glasgow, Scotland. The study described here was performed when he was an assistant professor in the Department of Information Systems in the W.P. Carey School of Business at Arizona State University, Tempe, AZ.

Weiguo Fan (wfan@vt.edu) is an associate professor of information systems and of computer science at Virginia Polytechnic Institute and State University, Blacksburg, VA.

José Robles-Flores (jrobles@esan.edu.pe) is an assistant professor in the School of Business at ESAN University, Lima, Perú. The study described here was performed while he was working on his doctoral dissertation in the Department of Information Systems in the W.P. Carey School of Business at Arizona State University, Tempe, AZ.



Figure 1: Code reviews go electronic and global.

DOI:10.1145/1378727.1378744

New collaboration tools allow geographically distributed software-development teams to boost the venerable concept of code review.

BY BERTRAND MEYER

Design and Code Reviews in the Age of the Internet

CODE REVIEWS ARE A standard practice of software engineering. Rather, they are a standard practice of the software engineering literature. Widely recommended, certainly, but how widely practiced, I am not sure.

In the development of EiffelStudio, a large integrated development environment (IDE), Eiffel Software has begun to apply code reviews with the added twist that the developers work on three continents. A distributed setup is increasingly common in the IT industry, though not always in such an extreme form. It naturally leads to distributing the reviews as well. This decision forced our development group to depart from the standard scheme described in the literature² and take a fresh look at the concept. Some of what initially appeared as constraints—

the impossibility of ever having all the people involved at the same time in the same room—turned out in practice to be beneficial, encouraging us to emphasize the written medium over verbal discussion, conduct much of the process prior to the actual review meeting, and take advantage of communication tools to allow several threads of discussion to proceed in parallel during the meeting itself. We also expanded our reviews, beyond just code, to cover design and specification as well. The process relies on modern, widely available communication and collaboration tools, most introduced over the past few years with considerable room for improvement. This article describes some of the lessons our team has learned with the hope they will be useful to other teams practicing distributed development.

Michael Fagan of IBM introduced the concept of “code inspections,” his original name, in a 1976 article.¹ Whether inspection or review, the idea is to examine some element of code in a meeting typically attended by perhaps eight people, with the aim of finding flaws or elements that should be improved. This is the meeting’s only goal. It is not intended to assess the code’s author, though in practice it is not easy to avoid doing so, especially when the manager is present; neither should it serve to correct deficiencies, only to uncover them.

The code and any associated elements are circulated a few days in advance. The meeting, which typically takes a few hours, includes the author, other developers competent to assess the code, a meeting chair (not the manager) who moderates the discussion, and a secretary who records it, producing a report with specific recommendations. Some time later, the author responds to the report by describing whether and how the recommendations have been carried out.

This is the basic idea behind a traditional code review and is often criticized on a variety of grounds. Advocates of extreme programming point

out that reviews may be superfluous if the project practices pair programming. Others note that when it comes to finding code flaws (such as a lurking buffer overflow) static analysis tools are more effective than human inspection. In any case, the process is time-consuming; most teams apply it on the entire code, as well as on samples. Still, code reviews remain an important tool in the battery of accepted “best practices” for improving software quality.

Our group has found that the exercise is indeed useful when adapted to the modern world of software development. The first extension is to include design and specification. Many recent

references concerning code reviews focus on detecting low-level code flaws, especially security risks. This is important but is increasingly a task for automated tools, not for humans. Our experience suggests that abstract program interface (API) design, architecture choices, and other specification and design issues are just as worthy of a reviewer’s time and play an increasingly important part in our reviews.

Among these traditional principles, one that should definitely be retained in the new distributed context is that reviews must focus on identifying deficiencies, not attempt to correct them. With the advent of better

software technology it may be tempting to couple review activities with actual changes in software repositories; one environment that supports Web-based review—Code Collaborator, www.smartbear.com—allows this by coupling the review tools with a configuration-management system. Such coupling may be risky; updating software—even for simple code changes with no effect on specification and design—is a delicate matter best performed in an environment free from the time pressure inherent in a review session.

Distributed Review

All the descriptions of code reviews I have read in the literature present a review as a physical meeting among people in the same room. This is hardly applicable to today’s increasingly dominant model of software development: distributed teams spread over many locations and time zones.³ At Eiffel Software, we were curious to see whether we could indeed apply the model; our first experience—still only a few months old and subject to refinement—suggests that thanks to the Internet and modern communications technology a distributed setup is less a hindrance than a benefit and that today’s technology provides a strong incentive to revive and expand the idea of the review.

Though the EiffelStudio development team includes members in California, China, Russia, and Western Europe, it still manages to have a weekly technical meeting, with some members agreeing to stay up late; for example, in the winter, 8 A.M. to 9 A.M. in California means 5 P.M. to 6 P.M. in Western Europe, 7 P.M. to 8 P.M. in Moscow, and midnight to 1 A.M. in Shanghai (see Figure 1). We devote every second or third such meeting to a design and code review.

Although many of the lessons we have learned should be valid for any team, some of the specifics of our reviews may influence our practice and conclusions. Our meetings, both ordinary ones and those devoted to reviews, last exactly one hour. We are strict on the time limit, obviously because it’s late at night for some team members but also to avoid wasting the time of a group of highly competent develop-

Figure 2: Excerpts from a chat session during a review (names blocked out).



ers. This constraint is an example of a limitation that has turned out to be an advantage, forcing us to organize both the reviews and other meetings carefully and professionally.


Almost all of our development is done in Eiffel; one aspect of this choice that influences the review process is that Eiffel applies the “seamless development” principle, treating specification, design, and analysis as a continuum rather than as a sequence of separate steps (using, for example, first UML then a programming language); the Eiffel language serves as the common notation throughout. This has naturally caused the extension of the traditional review to design reviews, an extension that may also be desirable for teams using other development languages and a less-seamless process. Another aspect that influences EiffelStudio development is that, since IDEs are our business, the tool we produce is also the tool we use (following the “eat your own dog food” principle); we again feel the results would not fundamentally change for other kinds of software development.

Distributed reviews need support from communication and collaboration tools; we essentially rely on four such tools:


Voice communication similar to a conference call. We started with Skype but now use it only as a backup; our primary VoIP solution is a tool called X-Lite; such technology choices are subject to reassessment as the tools evolve;

Written communication. We retain Skype’s chat mechanism, so a window available to all participants is active throughout the review;

Google Docs for shared documents. Providing a primitive Microsoft-Word-like editing framework, Google Docs works on the Web so several people are able to update a given document at the same time. The means of resolving editing conflicts is fine-grained: most changes go through, even if someone else is simultaneously modifying the document; only if two people are changing the very same words does the tool reject the requests. While not perfect, Google Docs is an effective tool for collaborative editing, with the advantage that text can be pasted into and from Microsoft Word documents with approximate preservation of format;



What is remarkable in the current setup is that we have not yet identified a need for specialized review software, being content enough with general-purpose widely available communication and collaboration tools.



WebEx sharing tool for sharing screens. We find this tool (one of several, including Adobe Connect, on the market) especially useful for running a demo of, say, a new proposal for a graphical-user-interface idea or other element developers might have just put together on a workstation; and

Wiki pages. The EiffelStudio community, involving both Eiffel Software developers and numerous outside contributors, has a Wiki-based site (dev.eiffel.com) with hundreds of documentation and discussion pages. The site is useful, although the Wiki mechanism, with its traditional edit cycle—start editor, make changes, update page, refresh—is less convenient than Google Docs for working on a common document during a meeting.

Among the ideas described here, the choice of tools is the most likely candidate for quick obsolescence. The technology is evolving so quickly that a year or two from publication the solutions might be fairly different. What is remarkable in the current setup is that we have not yet identified a need for specialized review software, being content enough with general-purpose widely available communication and collaboration tools.

Lessons Learned

Here are some of the lessons we have learned from our distributed code review:

First, *scripta manent*, or “prefer the written word.” We have found that a review works much better when it is organized around a document. The team members produce a shared document (currently in Google Docs) ahead of the meeting and update it in real time during the meeting.

The “unit of review” is a class or sometimes a small number of closely related classes. A week ahead of the review, the software’s author, following a standard structure described in the following sections, prepares the shared document with links to the actual code.

One way this process differs from a traditional review is a practice we had not planned for when we first put reviews in place but which quickly imposed itself through experience: Most of the work is done offline before the meeting. Our original thinking was

that it was preferable to limit the advance work and delay written comments until a couple of days before the meeting to avoid reviewers influencing one another too much. Experience showed that such concern was misplaced; interaction among reviewers, before, during, and after the meeting, is one of the most effective aspects of the process.

Prior to the meeting, the reviewers provide their comments on the review page (the shared document); the code author then responds just below the comments on the same page. The benefit of this approach is that it saves time. Before we systematized it we spent considerable time in the meeting on noncontroversial issues; in fact, our experience suggests that with a competent group of developers most comments and criticisms are readily accepted by the code's author. We should instead be spending our meeting time on the remaining points of disagreement. Otherwise we end up replaying the typical company board meeting as described in the opening chapter of C. Northcote Parkinson's *Parkinson's Law*.⁴ (The two items on the agenda are the color of bicycles for the mail messengers and whether to build a nuclear plant. Everyone has an opinion on colors, so the first item takes 59 minutes, ending with the decision to form a committee; the next decision is taken in one minute, with a resolution to let the CEO handle the matter.) Unlike this pattern, the verbal exchange in an effective review should target the issues that truly warrant discussion.

For an example of a document produced before and during one of our code reviews, see dev.eiffel.com/reviews/2008-02-sample.html, which gives a good idea of the process. For a more complete picture of what actually goes on during the meeting, also see the discussion extract from the chat window (names blocked out) in Figure 2.

Scope of Review

The standard review-page structure consists of nine sections dividing the set of software characteristics under review:

- ▶ Choice of abstractions;
- ▶ Other aspects of API design;
- ▶ Other aspects of architecture (such

as choice of client links and inheritance hierarchies);

- ▶ Contracts;
- ▶ Implementation, particularly the choice of data structures and algorithms;
- ▶ Programming style;
- ▶ Comments and documentation (including indexing/note clauses);
- ▶ Global comments; and
- ▶ Adherence to official coding practices.

The order of these sections goes from more high-level to more implementation-oriented. Note that the first four concern not just code but architecture and design as well:

- ▶ The choice of abstractions is the key issue of object-oriented design. Developers discuss whether a certain class is really justified or should have its functionalities merged with another's, or, conversely, whether an important potential class has been missed;
- ▶ API design is essential to the usability of software elements by other elements and as a consequence to reusability. We enforce systematic API design conventions through strong emphasis on consistency across the entire code base. This aspect of software development is particularly suitable for review; and
- ▶ Other architectural issues are also essential to good object-oriented development; the review process is useful (during both the preparatory phase and the meeting itself) to address such questions as whether a class should inherit from another or just be its client.

Algorithm design (the fifth section in the list) is another good candidate for discussion during the meeting. In our process, the lower-level sections—programming style, comments and documentation, global comments, and coding practices—are increasingly handled before the review meeting (in writing), enabling us to devote the meeting itself to the deeper, more delicate issues.

The division into nine sections and the distribution of work through written comments prior to the meeting and in-meeting verbal discussions yield the following benefits:

- ▶ The group saves time; precious personal interaction time is reserved only for the topics that really need it;
- ▶ Discussing issues in writing makes

it possible to include more thoughtful comments; participants can take care to express their observations, including criticism of design and implementation decisions and corresponding responses, more easily than through a verbal conversation alone;

- ▶ The written support allows editing and revision;
- ▶ A record is produced. Indeed, the review needs no secretary or the tedious process of writing minutes. The review page (in its final stage after joint editing) is the minutes;

▶ Verbal discussion time is much more interesting since it addresses issues of substance. The dirty secret of traditional code reviews is that most of the proceedings are boring to most participants, each of whom is typically interested in only a subset of all the items discussed. With an electronic meeting each participant resolves issues of specific concern in advance and in writing; the verbal discussion is devoted to the controversial and hence most interesting stuff; and


▶ In a group with contentious personalities, one may expect that expressing comments in writing will also help defuse tension. (I say "may expect" because I don't know this from experience; our developer group is not contentious.)

Through our electronic meetings, not just code reviews, another example has emerged of how constraints can be turned into benefits. Individually, most people (apart from, say, piano players) are most effective when doing one thing at a time, but collectively a group of humans is pretty good at multiplexing. When was the last time you spent an hour-long meeting willingly focused throughout on whatever issue was under discussion at the moment? Even the most attentive participants, disciplined to not let their minds wander off topic, react at different speeds; you may be thinking deeply about some previous item, while the agenda has moved on; you may be ahead of the game; or you may have something to say that complements the comments of the current speaker but do not want to interrupt her. This requires multithreading, but a traditional meeting is sequential. In our code reviews and other meetings we have learned to practice a kind of organic multithread-


ing: Someone may be talking; someone else may be writing a comment in the chat window (such as a program extract that illustrates a point under discussion or a qualification of what is being said); others may be updating the common document; and yet someone else may be preparing next week's document or a Wiki page at dev.eiffel.com. The dynamics of such meetings are amazing to behold, with threads progressing in parallel while everyone remains on target and alert.

Team distribution is a fact of life in today's software development and can be a great opportunity, as well as a challenge, to improve the engineering of software. I also practice distributed team development in an academic environment. ETH Zurich offers a course called Distributed and Outsourced Software Engineering, or DOSE, specifically devoted to the issues and techniques of distributed development. In fall 2007, it meant, for the first time, a cooperative project involving several universities. This was a trial run, and we are now expanding the experience; for details see se.ethz.ch/dose/. Participation is open to any interested university worldwide; the goal is to let students discover and confront the challenges of distributed development in the controlled environment of a university course.

Not all of our practice may be transposable to other contexts. The core EiffelStudio group includes only about 10 people who know each other well and have worked together for several years; we developed these techniques together, learning from our mistakes and benefiting from recent advances in the communication technology discussed here. But even if all the details of our experience cannot be generalized, distributed software development, and with it distributed reviews, are the way of the future. Even more so considering that the supporting technology is still in its infancy. As recently as 2006, most of the communication tools we use today did not exist; in 2003 none of them did. It is ironic now to recall the talks I heard over the years about "computer-supported cooperative work"—fascinating but remote from anything my colleagues or I could use. Suddenly comes the Web, VoIP solutions for common folk, shared editing



Interaction among reviewers, before, during, and after the meeting, has turned out to be one of the most effective aspects of the process.




tools, and new commercial offerings, and the gates to globalized cooperative development open without fanfare.

The tools are still fragile; we waste too much time on meta communication ("Can you hear me?," "Did Peter just disconnect?," "Bill, mute your microphone."), calls are cut off, and we lack a really good equivalent of a shared whiteboard. Other aspects of the process also still need improvement; for example, how can we make our review results seamlessly available as part of the EiffelStudio open-source development site based on Wiki pages? All this will be corrected in the next few years. I also hope that courses like DOSE and other academic efforts will enable the commercial world to understand better what makes collaborative development succeed or fail.

This is not just an academic issue. Eiffel Software's experience in collaborative development—whereby each meeting brings new insight—suggests that something fundamental has changed, mostly for the better, in the software-development process. As for code reviews, I do not expect ever again to get stuck for three hours in a windowless room with a half dozen other programmers poring over boring printouts.

Acknowledgment

I am grateful to the EiffelStudio developers for their creativity and the team spirit that enabled them collectively to uncover and apply the techniques discussed here. 

References

1. Fagan, M.E. Design and code inspections to reduce errors in program development. *IBM Systems Journal* 15, 3 (1976), 182–211; www.research.ibm.com/journal/sj/153/lbmsj1503C.pdf.
2. Ghezzi, C., Jazayeri, M., and Mandrioli, D. *Fundamentals of Software Engineering, Second Edition*. Prentice Hall, Upper Saddle River, NJ, 2002.
3. Meyer, B. and Piccioni, M. The allure and risks of a deployable software engineering project. In *Proceedings of the 21st IEEE-CS Conference on Software Engineering Education and Training* (Charleston, SC, Apr. 2008).
4. Parkinson, C. Northcote. *Parkinson's Law: The Pursuit of Progress*. John Murray, London, 1958.

Bertrand Meyer (Bertrand.Meyer@inf.ethz.ch) is a professor of software engineering at ETH Zurich, the Swiss Federal Institute of Technology, and chief architect of Eiffel Software, Santa Barbara, CA.

A guide to the tools and core technologies for merging information from disparate sources.

BY PHILIP A. BERNSTEIN AND LAURA M. HAAS

Information Integration in the Enterprise

LARGE ENTERPRISES SPEND a great deal of time and money on “information integration”—combining information from different sources into a unified format. Frequently cited as the biggest and most expensive challenge that information-technology shops face, information integration is thought to consume about 40% of their budget.^{4, 14, 16, 33, 36} Market-intelligence firm IDC estimates that the market for data integration and access software (which includes the key enabling technology for information integration) was about \$2.5 billion in 2007 and is expected to grow to \$3.8 billion in 2012, for an average annual growth rate of 8.7%.²⁰

Software purchases are only one part of the total expense. Integration activities cover any form of information reuse, such as moving data from one application’s database to another’s, translating messages for business-to-business e-commerce, and providing access to structured data and documents via a Web portal.

ILLUSTRATION BY LEANDER HERZOG



Beyond classical information-technology applications, information integration is also a large and growing part of science, engineering, and biomedical computing, as independent labs often need to use and combine each other's data.

Software vendors offer numerous tools to reduce the effort, and hence the cost, of integration and to improve the quality. Moreover, because information integration is a complex and multifaceted task, many of these tools are highly specialized. The resulting profusion of tools can be confusing. In this article, we try to clear up any confusion by:

- ▶ Exploring *an example* of a typical integration problem
- ▶ Describing *types of information integration tools* used in practice
- ▶ Reviewing *core technologies* that lie at the heart of integration tools
- ▶ Identifying *future trends*.

An Example

Consider a large auto manufacturer's support center that receives a flood of emails and service-call transcriptions every day. From any given text, company analysts can extract the type of car, the dealership, and whether the customer is pleased or annoyed with the service. But to truly understand the reasons for the customer's sentiment, the company also needs to know something about the dealership and

the transaction—information that is kept in a relational database.

Solving such an integration problem is an iterative process. The data must first be understood and then prepared for integration by means of "cleansing" and "standardization." Next, specifications are needed regarding what data should be integrated and how they are related. Finally, an integration program is generated and executed by some type of integration engine. The results are examined, and any anomalies must be resolved, which often requires returning to step one and studying the data.

Many technologies are needed to support this process. We introduce a few here and then describe them in greater depth, along with others, in subsequent sections.

The first step toward integrating the text and the relational data is to understand what transactions and other information they contain and how to relate that information to each dealership. The manufacturer next needs to decide how to represent the integrated information. A simple schema—auto model, customer, dealership, date sold, price, size of dealership, date of problem, problem type—might suffice (See Figure 1). But how should each field be represented, and where will the data come from? Let's assume that the auto model, customer, and dealership information will be extracted from the

textual complaint, as will the date of problem and problem type. The relational database has tables about dealerships and transactions that can provide the rest of the information: size of dealership, date sold, and price.

Next, programs are needed to extract structured information from the email or transcription text. These programs' outputs provide a schema for the text data—the "fields" that can be queried. Matching and mapping tools can be used to relate this derived schema to the target schema. Similarly, matching and mapping must be done for the relational schema. The dealership name extracted from the text can be connected to an entry in the dealership table, and the customer, auto model, and dealership to the transactions table, thus joining the two tables with the textual data.

Programs are needed to align data instances: because it is unlikely that the data formats of the extracted text are identical to those in the relational database, some data cleansing will be required. For example, dealership names may not exactly match. These data-integration programs must then be executed, often using a commercial integration product.

Types of Information-Integration Tools

A variety of architectural approaches can be used to solve problems like the

Figure 1. Annotators extract key information from email messages. This information is used to probe the relational source data to retrieve additional facts needed for the target schema.

TS: Oct. 28, 2007
 Sirs:
 My Galaxy's brakes are squealing after only 6 months! I purchased this clunker at Billboy in Oshkosh...
 Sincerely,
 John J. Jutt

Source Data and Schema

DEALERS

Dealership	DealerID	Size of Dealership	Owner	Annual Revenue
Oshkosh Billboy Ford	bb32	300 cars per month	Bill Boy	\$5M

TRANSACTIONS

TransID	Customer	DealerID	Auto Model	Date Sold	Price
1234	Jutt, John J.	bb32	Galaxy	4/21/07	5000

Target Schema

Auto Model	Customer	Dealership	Date Sold	Price	Size of Dealership	Problem Date	Problem Type
Galaxy	John J. Jutt	Oshkosh Billboy Ford	4/21/07	5000	300 cars per month	Oct. 28, 2007	brakes are squealing

one above. We summarize these approaches in this section, along with the general types of products used. For information on specific products, we refer the interested reader to IT research companies that publish comprehensive comparisons and to Web search engines (using the product categories we define here as keyword queries).

Data Warehouse Loading. A data warehouse is a database that consolidates data from multiple sources.⁷ For example, it may combine sales information from subsidiaries to give a sales picture for the whole company. Because subsidiaries have overlapping sets of customers and may have inconsistent information about a particular customer, data must be cleansed to reconcile such differences. Moreover, each subsidiary may have a database schema (that is, data representation) that differs from the warehouse schema. So each subsidiary's data has to be reshaped into the common warehouse schema.

Extract-Transform-Load (ETL) tools address this problem²¹ by simplifying the programming of scripts. An ETL tool typically includes a repertoire of cleansing operations (such as detection of approximate duplicates) and reshaping operations (such as Structured Query Language [SQL]-style operations to select, join, and sort data). The tool may also include scheduling functions to control periodic loading or refreshing of the data warehouse.

Some ETL tools are customized for master data management—that is, to produce a data warehouse that holds the master copy of critical enterprise reference data, such as information about customers or products. Master data is first integrated from multiple sources and then itself becomes the definitive source of that data for the enterprise. Master data-management tools sometimes include domain-specific functionality. For example, for customer or vendor information, they may have formats for name and address standardization and cleansing functions to validate and correct postal codes.

Virtual Data Integration. While warehouses materialize the integrated data, virtual data integration gives the illusion that data sources have been integrated without materializing the



Beyond classical information-technology applications, information integration is also a large and growing part of science, engineering, and biomedical computing, as independent labs often need to use and combine each other's data.



integrated view. Instead, it offers a mediated schema against which users can pose queries. The implementation, often called a query mediator³⁵ or enterprise-information integration (EII) system,^{16, 27} translates the user's query into queries on the data sources and integrates the result of those queries so that it appears to have come from a single integrated database. EII is still an emerging technology, currently less popular than data warehousing.

Although the databases cover related subject matter, they are heterogeneous in that they may use different database systems and structure the data using different schemas. An EII system might be used, for example, by a financial firm to prepare for each customer a statement of portfolio positions that combines information about his or her holdings from the local customer database with stock prices retrieved from an external source.

To cope with this heterogeneity in EII, a designer creates a mediated schema that covers the desired subject matter in the data sources and maps the data source schemas to the new mediated schema. Data cleansing and reshaping problems appear in the EII context, too. But the solutions are somewhat different in EII because data must be transformed as part of query processing rather than via the periodic batch process associated with loading a data warehouse.

EII products vary, depending on the types of data sources to be integrated. For example, some products focus on integrating SQL databases, some on integrating Web services, and some on integrating bioinformatics databases.

Message Mapping. Message-oriented middleware helps integrate independently developed applications by moving messages between them. If messages pass through a broker, the product is usually called an enterprise-application integration (EAI) system.¹ If a broker is avoided through all applications' use of the same protocol (for example, Web services), then the product is called an enterprise service bus. If the focus is on defining and controlling the order in which each application is invoked (as part of a multistep service), then the product is called a workflow system.

In addition to the protocol-transla-

tion and flow-control services provided by these products, message-translation services—which constitute another form of information integration—are also needed.

A typical message-translation scenario in e-commerce enables a small vendor (say, Pico) to offer its products through a large retail Web site (Goliath). When a customer buys one of Pico's products from Goliath, it sends an order message to Pico, which then has to translate that message into the format required by its order-processing system. A message-mapping tool can help Pico meet this challenge. Such a tool offers a graphical interface to define translation functions, which are then compiled into a program to perform the message translation. Similar mapping tools are used to help relate the schemas of the source databases to the target schema for ETL and EII and to generate the programs needed for data translation.

Object-to-Relational Mappers. Application programs today are typically written in an object-oriented language, but the data they access is usually stored in a relational database. While mapping applications to databases requires integration of the relational and application schemas, differences in schema constructs can make the mapping rather complicated. For example, there are many ways to map classes that are related by inheritance

into relational tables. To simplify the problem, an object-to-relational mapper offers a high-level language in which to define mappings.²³ The resulting mappings are then compiled into programs that translate queries and updates over the object-oriented interface into queries and updates on the relational database.

Document Management. Much of the information in an enterprise is contained in documents, such as text files, spreadsheets, and slide shows that contain interrelated information relevant to critical business functions—product designs, marketing plans, pricing, and development schedules, for example. To promote collaboration and avoid duplicated work in a large organization, this information needs to be integrated and published. Integration may simply involve making the documents available on a single Web page (such as a portal) or in a content-management system, possibly augmented with per-document annotations (on author and status, for example). Or integration may mean combining information from these documents into a new document, such as a financial analysis.

Whether or not the documents are collected in one store, they can be indexed to enable keyword search across the enterprise. In some applications, it is useful to extract structured information from documents, such as cus-

tomers name and address from email messages received by the customer-support team. The ability to extract structured information of this kind may also allow businesses to integrate unstructured documents with preexisting structured data. In the example above, the auto manufacturer wanted to link transactional information about purchases with emails about these purchases in order to enable better analysis of problem reports.

Portal Management. One way to integrate related information is simply to present it all, side-by-side, on the same screen. A portal is an entire Web site built with this type of integration in mind. For example, the home page of a financial services Web site typically presents market prices, business news, and analyses of recent trends. The person viewing it does the actual integration of the information.

Portal design requires a mixture of content management (to deal with documents and databases) and user-interaction technology (to present the information in useful and attractive ways). Sometimes these technologies are packaged together into a product for portal design.¹¹ But often they are selected piecemeal, based on the required functionality of the portal and the taste and experience of the developers who assemble it.

Core Technologies

Extensible Markup Language (XML). In any of the scenarios noted here, an integrated view of data from multiple sources must be created. Often any one of the sources will be incomplete with respect to that view, with each source missing some information that the others provide. In our example, the emails are unlikely to provide detailed information about the dealerships, while the relational data might not have the problem reports. In XML, a semi-structured format, each data element is tagged so that only elements whose values are known need to be included. This ability to handle variations in information content is driving EII systems to experiment with XML.²²

This flexibility makes XML an interesting format for integrating information across systems with differing representations of data. In some integration scenarios, it may not be

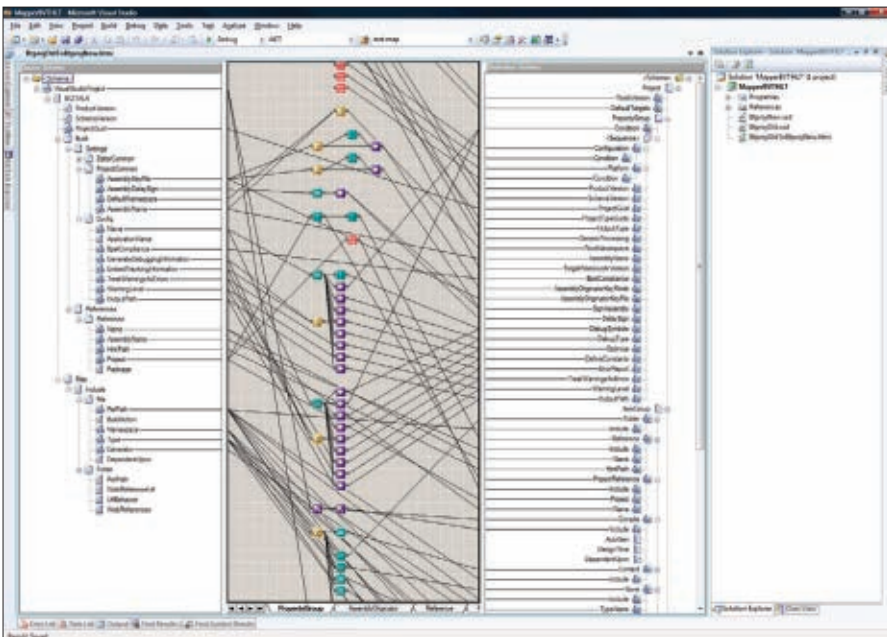


Figure 2. Screenshot of a mapping tool.

necessary to define a common schema—data from both sources can be merged into a single self-describing XML document—though in scenarios such as warehousing applications the transformation and fusing of the original data into a well-defined format is required. Still, its flexibility and the ubiquity of free parsers make XML attractive in scenarios with looser requirements, and it is increasingly being used for transferring data between systems and sometimes as a format for storing data as well.³


Schema Standards. It is easier to integrate data from different sources if they use the same schema. This consistency avoids the need to reformat the data before integrating it, and it also ensures that data from all of the sources have mutual meaning.

Even if sources do not conform to a common schema, each source may be able to relate its data to a common standard, either industry-wide or enterprise-specific. Thus two sources can be related by composing the two mappings that relate each of them to the standard. This approach only enables integration of information that appears in the standard, and because a standard is often a least common denominator, some information is lost in the composition.


There are many industry-wide schema standards.^{18, 28, 29} Some are oriented toward generic kinds of data, such as geographic information or software-engineering information. Others pertain to particular application domains such as computer-aided design, news stories, and medical billing.

When the schema standard is abstract and focuses on creating a taxonomy of terms, it is usually called an ontology. Ontologies are often used as controlled vocabularies—for example, in the biomedical domain—rather than as data formats.^{12, 13}

Data Cleansing. When the same or related information is described in multiple places (possibly within a single source), often some of the occurrences are inconsistent or just plain wrong—that is, “dirty.” They may be dirty because the data, such as inventory and purchase-order information about the same equipment, were independently obtained. Or they may simply have errors such as misspellings,



Information integration is a vibrant field powered not only by engineering innovation but also by evolution of the problem itself.



be missing recent changes, or be in a form that is inappropriate for a new use that will be made of it.

A typical initial step in information integration is to inspect each of the data sources—perhaps with the aid of data-profiling tools—for the purpose of identifying problematic data. Then a data-cleansing tool may be used to transform the data into a common standardized representation. A typical data-cleansing step, for example, might correct misspellings of street names or put all addresses in a common format.¹⁰ Often, data-profiling and -cleansing tools are packaged together as part of an ETL tool set.

One important type of data cleansing is entity resolution, or deduplication, which identifies and merges information from multiple sources that refer to the same entity. Mailing lists are a well-known application; we have all received duplicate mail solicitations with different spellings of our names or addresses. On the other hand, sometimes seeming “duplicates” are perfectly valid, because there really are two different persons with very similar names (John T. Jutt and his son John J. Jutt) living at the same address.

Many data-cleansing tools exist, based on different approaches or applied at different levels or scales. For individual fields, a common technique is edit-distance; two values are duplicates if changing a small number of characters transforms one value into the other. For records, the values of all fields have to be considered; more sophisticated systems look at groups of records and accumulate evidence over time as new data appears in the system.

Schema Mapping. A fundamental operation for all information-integration systems is identifying how a source database schema relates to the target integrated schema. Schema-mapping tools, which tackle this challenge, typically display three vertical panes (see Figure 2). The left and right panes show the two schemas to be mapped; the center pane is where the designer defines the mapping, usually by drawing lines between the appropriate parts of the schemas and annotating the lines with the required transformations. Some tools offer design assistance in generating these transfor-

mations, which are often complex.^{26,30}

Once the mapping is defined, most tools can generate a program to transform data conforming to the source schema into data conforming to the target schema.¹⁵ For an ETL engine, the tool might generate a script in the engine's scripting language. For an EII system, it might generate a query in the query language, such as SQL. For an EAI system, it might transform XML documents from a source-message format to that of the target. For an object-to-relational mapping system, it might generate a view that transforms rows into objects.²³

Schema Matching. Large schemas have several thousand elements, presenting a major problem for a schema-mapping tool. To map an element of Schema 1 into a plausible match in Schema 2, the designer may have to scroll through dozens of screens. To avoid this tedious process, the tool may offer a schema-matching algorithm,³¹ which uses heuristic or machine-learning techniques to find plausible matches based on whatever information it has available—for example, name similarity, data-type similarity, structure similarity, an externally supplied thesaurus, or a library of previously matched schemas. The human user must then validate the match.

Schema-matching algorithms do well at matching individual elements with somewhat similar names, such as `Salary_of_Employee` in Schema 1 and `EmpSal` in Schema 2, or when matching predefined synonyms, such as `Salary` and `Wages`. Some techniques leverage data values. For example, the algorithm might suggest a match between the element `Salary` of the source database and `Stpnd` of the target if they both have values of the same type within a certain numerical range.

But matching algorithms are ineffective when there are no hints to exploit. They cannot map an element called `PW` (that is, person's wages) to `EmpSal` when no data values are available; nor can they readily map combinations of elements, such as `Total_Price` in Schema 1 and `Quantity * Unit_Cost` in Schema 2. That is, these algorithms are helpful for avoiding tedious activities but not for solving subtle matching problems.

Keyword Search. Keyword search is

second nature to us all as a way to find information. A search engine accepts a user's keywords as input and returns a rank-ordered list of documents that is generated using a pre-built index and other information, such as anchor text and click-throughs.⁵ A less familiar view of search is as a form of integration—for example, when a Web search on a keyword yields an integrated list of pages from multiple Web sites. In more sophisticated scenarios, the documents to be searched reside in multiple repositories such as digital libraries or content stores, where it is not possible to build a single index. In such cases, federated search can be used to explore each store individually and merge the results.²⁴

While keyword search does integrate information, it does so "loosely." The results are often imprecise, incomplete, or even irrelevant. By contrast, integration of structured data via an ETL tool or a query mediator can create new types of records by correlating and merging information from different sources. The integration request has a precise semantics and the answer normally includes all possible relevant matches from these data sets, assuming that the source data and entity resolution are correct (both of these are big assumptions). Both precise and loose integration techniques have merit for different scenarios. Keyword search may even be used against structured data to get a quick feel for what is available and set the stage for more precise integration.

Information Extraction. Information extraction²⁵ is the broad term for a set of techniques that produce structured information from free-form text. Concepts of interest are extracted from document collections by employing a set of annotators, which may either be custom code or specially constructed extraction rules that are interpreted and executed by an information-extraction system. In some scenarios, when sufficient labeled training data is available, machine-learning techniques may also be employed.

Important tasks include named-entity recognition (to identify people, places, and companies, for example) and relationship extraction (such as customer's phone number or customer's address). When a text fragment is

recognized as a concept, that fact can be recorded by surrounding it with XML tags that identify the concept, by adding an entry in an index, or by copying the values into a relational table. The result is better-structured information that can more easily be combined with other information, thus aiding integration.

Dynamic Web Technologies. When a portal is used to integrate data, it usually needs to be dynamically generated from files and databases that reside on backend servers. The evolution of Web technologies has made such data access easier. Particularly helpful has been the advent of Web services and Really Simple Syndication (RSS) feeds, along with many sites offering their data in XML.⁶ Development technology has been evolving too, with rapid improvement of languages, runtime libraries, and graphical development frameworks for dynamic generation of Web pages.

One popular way to integrate dynamic content is a "mashup," which is a Web page that combines information and Web services. For example, because a service for displaying maps may offer two functions—one to display a map and another to add a glyph that marks a labeled position on the map—it could be used to create a mashup that displays a list of stores and their locations on the map. To reduce the programming effort of creating mashups, frameworks are now emerging that provide a layer of information integration analogous to EII systems, but which are tailored to the new "Web 2.0" environment.²

Future Trends

Today, every step of the information-integration process requires a good deal of manual intervention, which constitutes the main cost. Because integration steps are often complex, some human involvement seems unavoidable. Yet more automation is surely possible—for example, to explain the behavior of mappings, identify anomalous input data, and trace the source of query results.⁸ Researchers and product developers continue to explore ways to reduce human effort not only by improving the core technologies mentioned in this article and the integration tools that embody

them but also by trying to simplify the process through better integration of the tools themselves.


Information integration is currently a brittle process; changing the structure of just one data source can force an integration redesign. This problem of schema evolution³² has received much attention from researchers; but surprisingly few commercial tools that might reduce the cost of integration are available to address the problem. Another cause of brittleness, and another topic of research,⁹ arises from the complex rules for handling the inconsistencies and incompleteness of different sources. One possible approach, for example, is to offer a tool that suggests minimal changes to source data, thereby eliminating many of the unanticipated inconsistencies.

Most past work has focused on the problems of information-technology shops, where the goal of integration is usually known at the outset of a project. But some recent work addresses problems in other domains, notably science, engineering, and personal-information management. In these domains, information integration is often an exploratory activity in which a user integrates some information, evaluates the result, and consequently identifies additional information to integrate. In this scenario, called “dataspaces,”¹⁷ finding the right data sources is important, as is automated tracking of how the integrated data was derived, called its “provenance.”³⁴ Semantic technologies such as ontologies and logic-based reasoning engines may also help with the integration task.¹⁹

Information integration is a vibrant field powered not only by engineering innovation but also by evolution of the problem itself. Initially, information integration was stimulated by the needs of enterprises; for the last decade, it has also been driven by the desire to integrate the vast collection of data available on the Web. Recent trends—the continual improvement of Web-based search, the proliferation of hosted applications, cloud storage, Web-based integration services, and open interfaces to Web applications (such as social networks), among others—present even more challenges to the field. Information integration will

keep large numbers of software engineers and computer-science researchers busy for a long time to come.

Acknowledgments

We are grateful to Denise Draper, Alon Halevy, Mauricio Hernández, David Maier, Sergey Melnik, Sriram Raghavan, and the anonymous referees for many suggested improvements. 

References

- Alonso, G., Casati, F., Kuno, H.A., and Machiraju, V. *Web Services—Concepts, Architectures and Applications*. Springer, 2004.
- Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, Y-H, Simmen, D.E., and Singh, A. DAMIA—A data mashup fabric for intranet applications. *VLDB Conference* (2007), 1370–1373.
- Babcock, C. XML plays big integration role. *InformationWeek* (May 24, 2004); www.informationweek.com/story/showArticle.jhtml?articleID=20900153.
- Bernstein, P.A. and Melnik, S. Model management 2.0: Manipulating richer mappings. In *Proceedings of the ACM SIGMOD Conference*, 2007, 1–12.
- Brin, S. and Page, L. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks* 30, 1–7 (1998), 107–117.
- Carey, M.J. Data delivery in a service-oriented world: The BEA AquaLogic data services platform. In *Proceedings of the ACM SIGMOD Conference* (2006), 695–705.
- Chaudhuri, S. and Dayal, U. An overview of data warehousing and OLAP technology. *ACM SIGMOD Record* 26, 1 (1997), 65–74.
- Chiticariu, L. and Tan, W.C. Debugging schema mappings with routes. *VLDB Conference* (2006), 79–90.
- Chomicki, J. Consistent query answering: Five easy pieces. In *Proceedings of the International Conference on Database Theory* (2007), 1–17.
- Dasu, T., and Johnson, T. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.
- Firestone, J.M. *Enterprise Information Portals and Knowledge Management*. Butterworth-Heinemann (Elsevier Science, KMCI Press), 2003.
- Foundational Model of Anatomy, Structural Informatics Group, University of Washington; <http://sig.biostr.washington.edu/projects/fm/>
- Gene Ontology; <http://www.geneontology.org/>.
- Haas, L.M. Beauty and the beast: The theory and practice of information integration. *International Conference on Database Theory* (2007), 28–43.
- Haas, L.M., Hernández, M.A., Ho, H., Popa, L., and Roth, M. Clio grows up: From research prototype to industrial tool. In *Proceedings of the ACM SIGMOD Conference* (2005), 805–810.
- Halevy, A.Y., Ashish, N., Bitton, D., Carey, M.J., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V. Enterprise information integration: Successes, challenges, and controversies. In *Proceedings of the ACM SIGMOD Conference* (2005), 778–787.
- Halevy, A.Y., Franklin, M.J., and Maier, D. Principles of dataspace systems. *ACM Symposium on Principles of Database Systems* (2006), 1–9.
- Health Level Seven; <http://www.hl7.org/>.
- Hepp, M., De Leenheer, P., de Moor, A., and Sure, Y. (Eds.). *Ontology management: Semantic web, semantic web services, and business applications*. Vol. 7 of series Semantic Web And Beyond. Springer, 2008.
- IDC. *Worldwide Data Integration and Access Software 2008–2012 Forecast*. Doc No. 211636 (Apr. 2008).
- Kimball, R. and Caserta, J. *The Data Warehouse ETL Toolkit*. Wiley and Sons, 2004.
- Ludascher, B., Papakonstantinou, Y., and Velikhov, P. Navigation-driven evaluation of virtual mediated views. *Extending Database Technology* (2000), 150–165.
- Melnik, S., Adya, A., and Bernstein, P.A. Compiling mappings to bridge applications and databases. In *Proceedings of the ACM SIGMOD Conference* (2007), 461–472.
- Meng, W., Yu, C., and Liu, K. Building efficient and

effective metasearch engines. *ACM Computing Surveys* 34, 1 (2002), 48–89.

- McCallum, A. Information extraction: Distilling structured data from unstructured text. *ACM Queue* 3, 9 (Nov. 2005).
- Miller, R.J., Haas, L.M., and Hernández, M.A. Schema mapping as query discovery. *VLDB Conference* (2000), 77–88.
- Morgenthal, J.P. *Enterprise Information Integration: A Pragmatic Approach*. Lulu.com, 2005.
- OASIS standards; www.oasis-open.org/specs/.
- OMG Specifications; www.omg.org/technology/documents/modeling_spec_catalog.htm.
- Popa, L., Velegrakis, Y., Miller, R.J., Hernández, M.A., and Fagin, R. Translating Web data. *VLDB Conference* (2002), 598–609.
- Rahm, E. and Bernstein, P.A. A survey of approaches to automatic schema matching. *VLDB Journal* 10, 4 (2001), 334–350.
- Roddick, J.F. and de Vries, D. Reduce, reuse, recycle: Practical approaches to schema integration, evolution, and versioning. *Advances in Conceptual Modeling—Theory and Practice, Lecture Notes in Computer Science*, 4231. Springer, 2006.
- Smith, M. Toward enterprise information integration. *Softwaremag.com* (Mar. 2007); www.softwaremag.com/L.cfm?Doc=1022-3/2007.
- Tan, W-C. Provenance in databases: past, current, and future. *IEEE Data Eng. Bulletin* 30, 4 (2007), 3–12.
- Wiederhold, G. Mediators in the architecture of future information systems. *IEEE Computer* 25, 3 (1992), 38–49.
- Workshop on Information Integration, October 2006; <http://db.cs.upenn.edu/iworkshop/postworkshop/index.htm>.

Philip A. Bernstein (philbe@microsoft.com) is a principal researcher in the database group of Microsoft Research in Redmond, WA.

Laura M. Haas (laura@almaden.ibm.com) is an IBM distinguished engineer and director of computer science at the IBM Almaden Research Center in San Jose, CA.

© 2008 ACM 0001-0782/08/0900 \$5.00

The Best Place to Find the Perfect Job... Is Just a Click Away!

No need to get lost on commercial job boards.
The ACM Career & Job Center is tailored specifically for you.

JOBSEEKERS

- ❖ Manage your job search
- ❖ Access hundreds of corporate job postings
- ❖ Post an anonymous resume
- ❖ Advanced Job Alert system

EMPLOYERS

- ❖ Quickly post job openings
- ❖ Manage your online recruiting efforts
- ❖ Advanced resume searching capabilities
- ❖ Reach targeted & qualified candidates

NEVER LET A JOB OPPORTUNITY PASS YOU BY!
START YOUR JOB SEARCH TODAY!

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



Association for
Computing Machinery

Advancing Computing as a Science & Profession

POWERED BY  **JOBTARGET**



research highlights

P. 82

**Technical
Perspective
Transactional
Memory in the
Operating System**

By Mark Moir

P. 83

**TxLinux and MetaTM:
Transactional Memory and
the Operating System**

By Christopher J. Rossbach, Hany E. Ramadan, Owen S. Hofmann,
Donald E. Porter, Aditya Bhandari, and Emmett Witchel

P. 92

**Technical
Perspective
Distributing
Your Data and
Having It, Too**

By Hagit Attiya

P. 93

**Distributed Selection:
A Missing Piece
of Data Aggregation**

By Fabian Kuhn, Thomas Locher, and Roger Wattenhofer

Technical Perspective

Transactional Memory in the Operating System

By Mark Moir

THE LONG TRADITION of building ever-faster processors is ending, with the computer industry instead putting more processing “cores” on each processor chip. Therefore, to continue to benefit from advances in technology, applications must increasingly be able to perform useful work on multiple cores at the same time.

To use multiple cores concurrently, programmers must not only identify independent pieces of a task that can be executed in parallel, but also coordinate their execution, managing communication and synchronization between them. A program with a communication or synchronization *bottleneck* will be unable to take full advantage of the available cores. *Scalable* programs that avoid such bottlenecks are surprisingly difficult to construct.

The complex interactions between concurrently executing pieces of code are often managed by the careful use of *locks* to ensure a *critical section* of code on one core executes “atomically,” that is, without interference from other cores. However, it is challenging to avoid synchronization bottlenecks with this approach, and in many cases the overhead of such bottlenecks negates the advantage gained by running pieces of the application concurrently.

Research in Transactional Memory (TM) has allowed programmers to express that a critical section of code should be executed atomically, without specifying *how* this should be achieved. The term “Transactional Memory” originated with seminal work by Herlihy and Moss (ISCA ‘93), in which they proposed to apply the idea of “transactions” (already popular in database systems) to computer memory.

Numerous variations on this theme have emerged in the literature, with some proposing TM be supported entirely in hardware, others entirely in software, and yet others using some combination of the two. Some propose extensions to programming lan-

guages, implemented with a combination of compiler support and runtime libraries, while others propose library interfaces, and yet others propose instruction set extensions that are programmed directly.

TM is generally “optimistic.” Rather than pessimistically *preventing* critical sections from executing concurrently just in case they interfere with others, as locks do, TM allows them to execute concurrently by default. The TM system monitors concurrent critical sections to detect any “conflict” between them. When conflicts do arise, the TM system forces one or more of the critical sections to either wait or roll back, so that no interference occurs. This is done automatically by the TM system.

To date, most research has focused on using TM in user programs, and emerging TM proposals have mostly been evaluated using rather artificial toy applications and benchmarks designed to explore the various trade-offs that arise in TM design.

In contrast, the research described here by Rossbach et al. explores the use of TM in an operating system. Such research is important for a number of reasons. First, it provides significant value to the TM research community by creating TM-based programs (operating systems) that are larger, more complex, and more realistic than the mostly artificial workloads used for evaluating TM systems so far. Second, TM has the potential to improve the performance and scalability of the operating system itself, even while making it simpler. Because all user programs depend on the operating system, its scalability and correctness is critical to the success of multicore systems.


TM research for user programs will not necessarily result in the best TM support for operating systems. On the one hand, the operating system faces some challenges that user programs do not. In particular, the operating system must perform reliably while running a

variety of user programs simultaneously and thus cannot exploit knowledge of specific ones to improve performance and scalability. Furthermore, operating systems employ a wider variety of synchronization mechanisms than typical user programs do, and must also deal with a variety of complicated issues on behalf of user programs, such as access to low-level system devices.

However, the operating system can exploit its privileged role in the system in ways that user programs cannot. For example, for security reasons, a user program cannot prevent the operating system from interrupting it, while the operating system has full control of the cores and can thus prevent interruption while finishing an important task.

The authors built two operating systems that exploit TM. They first attempted to use transactions directly instead of existing synchronization mechanisms. This approach entailed a number of challenges, particularly related to the fact that locks are used for purposes other than preventing memory conflicts between critical sections, such as protecting system devices for input and output (I/O). TM systems are not yet mature enough to support such functionality, and the lessons learned here are valuable for TM researchers.

The authors then took a more pragmatic approach: they retained existing synchronization mechanisms, and used TM to attempt to improve their performance and scalability. This way, they could use the existing locks when necessary—for example, for I/O—while also exploiting TM in other cases to allow parallelism that would previously have been impossible. Given the thousands of person-years spent carefully engineering modern operating systems, and the limited nature of the first TM systems, this is likely to be the most practical approach to exploiting TM in operating systems for some time.

By exploring both approaches, the authors not only contribute to our understanding of how TM can be used in large and realistic code bases, but also provide valuable insight into what additional TM features may be useful or necessary for optimal support of such systems in the longer term. 

Mark Moir (Mark.Moir@sun.com) is a senior staff engineer at Sun Microsystems Laboratories, Burlington, MA.

TxLinux and MetaTM: Transactional Memory and the Operating System

By Christopher J. Rossbach, Hany E. Ramadan, Owen S. Hofmann, Donald E. Porter,
Aditya Bhandari, and Emmett Witchel

Abstract

TxLinux is the first operating system to use hardware transactional memory (HTM) as a synchronization primitive, and the first to manage HTM in the scheduler. TxLinux, which is a modification of Linux, is the first real-scale benchmark for transactional memory (TM). MetaTM is a modification of the x86 architecture that supports HTM in general and TxLinux specifically.

This paper describes and measures TxLinux and MetaTM, the HTM model that supports it. TxLinux greatly benefits from a new primitive, called the cooperative transactional spinlock (*cxspinlock*) that allows locks and transactions to protect the same data while maintaining the advantages of both synchronization primitives. Integrating the TxLinux scheduler with the MetaTM's architectural support for HTM eliminates priority inversion for several real-world benchmarks.

1. INTRODUCTION

To increase performance, hardware manufacturers have turned away from scaling clock speed and are focusing on scaling the number of cores on a chip. Increasing performance on new hardware will require finding ways to take advantage of the parallelism made available by multiple hardware processing contexts—a burden placed directly on the software programmer. New generations of hardware will not increase the performance of user applications unless something is done to make concurrent programming easier, so the need for accessible approaches to parallel programming is increasingly urgent.

The current approach to achieving concurrency using parallel programming relies heavily on threading. Multiple sequential flows of control (threads) execute at the same time using locks to protect critical sections. Locks guarantee mutually exclusive access to shared resources. Unfortunately, parallel programming using threads and locks remains quite difficult, even for experienced programmers. Locks suffer from a number of well-known and long-lamented problems such as deadlock, convoys, and priority inversion; they compose poorly and require complex ordering disciplines to coordinate the use of multiple locks. There is also an unattractive performance-complexity trade-off associated with locks. Coarse-grain locking is simple to reason about but sacrifices concurrent performance. Fine-grain locking may enable high performance, but it makes code more complex, harder to maintain because it is dependent

on invariants that are difficult to express or enforce. TM has been the focus of much recent research attention as a technique that can provide the performance of fine-grain locking with the code complexity of coarse-grain locking.

TM is a programming model that can greatly simplify parallel programming. A programmer demarcates critical sections that may access shared data as *transactions*, which are sequences of memory operations that either execute completely (commit) or have no effect (abort). The system is responsible for ensuring that transactions execute *atomically* (either completely or not at all), and in *isolation*, meaning that a transaction cannot see the effects of other active transactions, and its own operations are not visible in the system until it commits. While transactions provide the abstraction of completely serial execution of critical section, the system actually executes them optimistically, allowing multiple transactions to proceed concurrently, as long as atomicity and isolation are not violated. The programmer benefits because the system provides atomicity: reasoning about partial failures in critical sections is no longer necessary. Because transactions can be composed, and do not suffer from deadlock, programmers can freely compose thread-safe libraries based on transactions.

HTM provides an efficient hardware implementation of TM that is appropriate for use in an OS. Operating systems benefit from using TM because TM provides a simpler programming model than locks. For instance, operating system has locking disciplines that specify the order in which locks must be acquired to avoid deadlock. These disciplines become complex over time and are difficult for programmers to master; transactions require no ordering disciplines. Because many applications spend a significant fraction of their runtime in the kernel (by making system calls, e.g., to read and write files), another benefit of TM in the OS is increased performance for user programs without having to modify or recompile them.

However, management and support of HTM in an operating system requires innovation both in the architecture and the operating system. Transactions cannot simply replace or eliminate locks in an operating system for two main reasons. The first is that many kernel critical sections perform I/O, actually changing the state of devices like the disk or network card. I/O is a problem for TM because TM systems assume that if a conflict occurs, one transaction can be aborted, rolled back to its start, and re-executed. However, when the OS performs I/O it actually changes the

state of a device (e.g., by writing data to the network). Most devices cannot revert to a previous state once a write operation completes, so a transaction that performs I/O cannot be rolled back and re-executed. The second reason is that some kernel critical sections are highly contended and currently locks are more efficient than transactions for highly contended critical sections. Under contention, the optimism of transactions is unwarranted and the rollbacks and back-off performed by the TM system can significantly reduce performance.

The *cxspinlock* (cooperative transactional spinlock) is a new primitive that addresses the problem of I/O in transactions, allowing locks and transactions to work together to protect the same data while maintaining both of their advantages. Previous HTM proposals require every execution of a critical section to be protected by either a lock or a transaction, while *cxspinlocks* allow a critical section or a data structure accessed from different critical sections to sometimes be protected by a lock and sometimes by a transaction. *Cxspinlocks* dynamically and automatically choose between locks and transactions. *Cxspinlocks* attempt to execute critical sections as transactions by default, but when the processor detects an I/O attempt, the transactions are rolled back, and the *cxspinlock* will ensure that the thread re-executes the critical section exclusively, blocking other transactional and non-transactional threads. Additionally, *cxspinlocks* provide a convenient API for converting lock-based code to use transactions.

HTM enables a solution to the long-standing problem of priority inversion due to locks. Priority inversion occurs when a high priority thread waits for a lock held by a low priority thread. We demonstrate the modifications necessary in the TxLinux scheduler and the TM hardware to nearly eliminate priority and policy inversion. Moreover, the OS can improve its scheduling algorithms to help manage high contention by leveraging a thread's transaction history to calculate the thread's dynamic priority or de-schedule conflicting threads.

This paper makes the following contributions:

1. Creation of a transactional operating system, TxLinux, based on the Linux kernel. TxLinux is among the largest real-life programs that use HTM, and the first to use HTM inside a kernel.
2. Novel mechanism for cooperation between transactional and lock-based synchronization of a critical region. The cooperative transactional spinlock (*cxspinlock*) can be called from a transactional or non-transactional thread, and it exploits the greater parallelism enabled by transactions.
3. Novel mechanism for handling I/O within transactions: transactions that perform I/O are restarted by the hardware and acquire a conventional lock in software.
4. HTM mechanism to nearly eliminate priority inversion.

2. HTM PRIMER

This section provides background on parallel programming with locks and gives an overview of programming with HTM.

2.1. Threads, synchronization, and locks

Current parallel programming practices rely heavily on the *thread* abstraction. A thread is a sequential flow of control, with a private program counter and call stack. Multiple threads may share a single address space, allowing them to communicate through memory using shared variables. Threads make it possible for a single logical task to take advantage of multiple hardware instruction processors, for example, by moving subsets of the task to different processing contexts and executing them in parallel. Threads allow an application to remain responsive to users or get other work done while waiting for input from a slow device such as a disk drive or a human beings. Multiple processors are the parallel computing resource at the hardware level, multiple threads are the parallel computing resource at the operating system level.

Threads require synchronization when sharing data or communicating through memory to avoid race conditions. A race condition occurs when threads access the same data structure concurrently in a way that violates the invariants of the data structure. For instance, a race condition between two threads inserting into a linked list could create a loop in the list. Synchronization is the coordination that eliminates race conditions and maintains data structure invariants (like every list is null terminated). *Locks* allow threads to synchronize concurrent accesses to a data structure. A lock protects a data structure by enforcing *mutual exclusion*, ensuring that only one thread can access that data structure at a time. When a thread has exclusive access to a data structure, it is guaranteed not to see partially completed changes made by other threads. Locks thus help maintain consistency over shared variables and resources.

Locks introduce many challenges into the programming model, such as deadlock and priority inversion. Most importantly though, they are often a mismatch for the programmer's real needs and intent: a critical section expresses a consistency constraint, while a lock provides exclusion. Ultimately, when a programmer encloses a set of instructions in a critical section, it represents the assessment that those instructions must be executed atomically (either completely, or not at all), and in isolation (without visible partial updates) in order to preserve the consistency of the data manipulated by the critical section. HTM provides hardware and software support for precisely that abstraction: atomic, isolated execution of critical sections. Locks can provide that abstraction conservatively by ensuring that no two threads are ever executing in the critical section concurrently. By contrast, TM provides this abstraction optimistically, by allowing concurrent execution of critical sections, detecting violations of isolation dynamically, and restarting one or more transactions in response, reverting state changes done in a transaction if the transaction does not commit. The result is a globally consistent order of transactions.

There are many lock variants, like reader/writer locks and sequence locks. These lock variants reduce the amount of exclusion for a given critical section which can improve performance by allowing more threads to concurrently execute a critical region. However these variants can be used

only in particular situations, such as when a particular critical region only reads a data structure. While lock variations can reduce the performance problems of locks, they do not reduce complexity, and in fact increase complexity as developers and code maintainers must continue to reason about whether a particular lock variation is still safe in a particular critical region.

2.2. Synchronization with transactions

HTM is a replacement for synchronization primitives such as spinlocks and sequence locks. Transactions are simpler to reason about than locks. They improve performance by eliminating lock variables and the coherence cache misses associated with them, and they improve scalability by allowing concurrent execution of threads that do not attempt to update the same data.

Transactions compose a thread executing a transaction can call into a module that starts another transaction. The second transaction *nests* inside the first. In contrast, most lock implementations do not compose. If one function takes a lock and then calls another function which eventually tries to take the same lock, the thread will deadlock. Research on transaction nesting semantics is an active area,^{13,15,16} but flat nesting, in which all nested transactions are subsumed by the outermost transaction, is easy to implement. MetaTM uses flat nesting, but all patterns of transaction nesting are free from deadlock and livelock.

HTM designs share a few key high level features: primitives for managing transactions, mechanisms for detecting conflicts between transactions, called *conflict detection*, and mechanisms for handling conflicts when they occur, or *contention management*.

The table here provides an HTM glossary, defining important concepts, and listing the primitives MetaTM adds to the x86 ISA. The machine instructions not shown in italics are those which are generic to any HTM design. Those shown in italics are specific to MetaTM. The **xbegin** and **xend** instructions start and end transactions, respectively. Starting a transaction causes the hardware to enforce isolation for reads and writes to memory until the transaction commits; the updates become visible to the rest of the system on commit. The **xretry** instruction provides a mechanism for explicit restart.

The set of memory locations read and written during a transaction are called its *read-set* and *write-set*, respectively. A *conflict* occurs between two transactions when there is a non-empty intersection between the write-set of one transaction and the union of the read- and write-sets of another transaction. Informally, a conflict occurs if two transactions access the same location and at least one of those accesses is a write-set.

When two transactions conflict, one of those transactions will proceed, while the other will be selected to discard its changes and restart execution at **xbegin**: implementation of a policy to choose the losing transaction is the responsibility of a *contention manager*. In MetaTM, the contention manager is implemented in hardware. The policies underlying contention management decisions can have a first-order impact on performance.²⁴ Advanced issues in contention management include *asymmetric conflicts*, in which one of

Hardware TM concepts in MetaTM.

Primitive	Definition
xbegin	Instruction to begin a transaction.
xend	Instruction to commit a transaction.
xretry	Instruction to restart a transaction.
<i>xgettxid</i>	Instruction to get the current transaction identifier, which is 0 if there is no currently active transaction.
<i>xpush</i>	Instruction to save transaction state and suspend current transaction. Used on receiving an interrupt.
<i>xpop</i>	Instruction to restore previously saved transaction state and continue xpushed transaction. Used on an interrupt return.
<i>xtest</i>	If the value of the variable equals the argument, enter the variable into the transaction read-set (if a transaction exists) and return true. Otherwise, return false, and do not enter the variable in the read-set.
<i>xcas</i>	A compare and swap instruction that subjects non-transactional threads to contention manager policy.
Conflict	One transactional thread writes an address that is read or written by another transactional thread.
Asymmetric conflict	A non-transactional thread reads (writes) an address written (read or written) by a transactional thread. (Also known as a violation of <i>strong</i> isolation.)
Contention	Multiple threads attempt to acquire the same resource, e.g., access to a particular data structure.
Transaction status word	Encodes information about the current transaction, including reason for most recent restart. Returned from xbegin .

the conflicting accesses is performed by a thread outside a transaction.

3. HTM AND OPERATING SYSTEMS

This section discusses motivation for using TM for synchronization in an operation system, and considers the most common approach to changing lock-based programs to use transactions.

3.1. Why use HTM in an operating system?

Modern operating systems use all available hardware processors in parallel, multiplexing the finite resources of the hardware among many user processes concurrently. The OS delegates critical tasks such as servicing network connections or swapping out unused pages to independent kernel threads that are scheduled intermittently. A process is one or more kernel threads, and each kernel threads is scheduled directly by the OS scheduler.

The result of aggressive parallelization of OS work is substantial sharing of kernel data structures across multiple threads within the kernel itself. Tasks that appear unrelated can create complex synchronization in the OS. Consider, for example, the code in Figure 1, which is a simplification of the Linux file system's `dparent_notify` function. This function is invoked to update the parent directory's modify time when a file is accessed, updated, or deleted. If two separate user processes write to different files in the same directory concurrently, two kernel threads can call this function

at the same time to update the parent directory modify time, which will manifest as contention not just for the `dentry->d_lock` but for the parent directory's `p->d_lock`, and `p->d_count`, as well as the `dcache->lock`). While an OS provides programmers with the abstraction of a single sequential operation involving a single thread of control, all of these threads coexist in the kernel. Even when the OS manages access to *different* files for *different* programs, resources can be used concurrently as a result, and the OS must synchronize its own accesses to ensure the integrity of the data structures involved.

To maintain good performance in the presence of such sharing patterns, many OSes have required great programmer effort to make synchronization fine-grained—i.e., locks only protect the minimum possible data. However, synchronization makes OS programming and maintenance difficult. In one comprehensive study of Linux bugs, 346 of 1025 bugs (34%) involved synchronization, and another study⁷ found four confirmed and eight unconfirmed deadlock bugs in the Linux 2.5 kernel. The complexity of synchronization is evident in the Linux source file `mm/filemap.c` that has a 50 line comment on the top of the file describing the lock ordering used in the file. The comment describes locks used at a calling depth of four from functions in the file. Locking is not composable; a component must know about the locks taken by another component in order to avoid deadlock.

TM can help reduce the complexity of synchronization in contexts like the `dparent_notify` function. Because multiple locks are involved, the OS must follow a locking ordering discipline to avoid deadlock, which would be unnecessary with TM. The fine-grain locking illustrated by `dparent_notify`'s release of the `dentry->d_lock` and subsequent acquisition of the `p->d_lock` and `dcache_lock` could be elided with transactions. If the function is called with *different* parent directories, the lock-based code still forces some serialization because of the `dcache->lock`. However, transactions can allow concurrent executions of critical sections when they do not contend for the

same data. TM is more modular than locks and can provide greater concurrency with simpler/coarser locks; operating systems can benefit.

3.2. Converting Linux to TxLinux-SS

Figure 1 also illustrates the most common paradigm for introducing transactions into a lock-based program: mapping lock acquires and releases to transaction begin and end, respectively. This was the first approach taken to using transactions in Linux, called TxLinux-SS. Linux features over 2000 static instances of spinlocks, and most of the transactions in TxLinux-SS result from converted spinlocks. TxLinux-SS also converts reader/writer spinlock variants and se-qllocks to transactions. Based on profiling data collected from the Syncchar tool,¹⁸ the locks used in nine subsystems were converted to use transactions. TxLinux-SS took six developers a year to create, and ultimately converted approximately 30% of the dynamic locking calls in Linux (in our benchmarks) to use transactions.

The TxLinux-SS conversion of the kernel exposes several serious challenges that prevent rote conversion of a lock-based operating system like Linux to use transactions, including idiosyncratic use of lock functions, control flow that is difficult to follow because of heavy use of function pointers, and most importantly, I/O. In order to ensure isolation, HTM systems must be able to roll back the effects of a transaction that has lost a conflict. However, HTM can only roll back processor state and the contents of physical memory. The effects of device I/O, on the other hand, cannot be rolled back, and executing I/O operations as part of a transaction can break the atomicity and isolation that transactional systems are designed to guarantee. This is known as the “output commit problem.”⁶ A computer system cannot un-launch missiles.

If the `dentry_iput` function in Figure 1, performs I/O, the TxLinux-SS transactionalization of the kernel will not function correctly if the transaction aborts. TM alone is insufficient to meet all the synchronization needs of an

Figure 1: Three adapted versions of the Linux file system `dparent_notify()` function, which handles update of a parent directory when a file is accessed, updated, or deleted. The leftmost version uses locks, the middle version uses bare transactions and corresponds to the code in TxLinux-SS, and the rightmost version uses `cxspinlocks`, corresponding to TxLinux-CX. Note that the `dentry_iput` function can do I/O.

```

void
dnotify_parent(dentry_t *dentry,
               ulong evt) {
    spin_lock(&dentry->d_lock);
    dentry_t * p = dentry->d_parent;
    dget(p);
    spin_unlock(&dentry->d_lock);
    inode_dir_notify(p->d_inode, evt);
    spin_lock(&dcache_lock);
    if(!(--p->d_count)) {
        spin_lock(&p->d_lock);
        dentry_iput(p);
        d_free(p);
        spin_unlock(&p->d_lock);
    }
    spin_unlock(&dcache_lock);
}

void
dnotify_parent(dentry_t *dentry,
               ulong evt) {
    xbegin;
    dentry_t *p = dentry->d_parent;
    dget(p);
    inode_dir_notify(p->d_inode, evt);
    if(!(--p->d_count)) {
        dentry_iput(p);
        d_free(p);
    }
    xend;
}

void
dnotify_parent(dentry_t *dentry,
               ulong evt) {
    cx_optimistic(&dentry->d_lock);
    dentry_t * p = dentry->d_parent;
    dget(p);
    cx_end(&dentry->d_lock);
    inode_dir_notify(p->d_inode, evt);
    cx_optimistic(&dcache_lock);
    if(!(--p->d_count)) {
        cx_optimistic(&p->d_lock);
        dentry_iput(p);
        d_free(p);
        cx_end(&p->d_lock);
    }
    cx_end(&dcache_lock);
}

```


operating system. Critical sections protected by locks will not restart and so may freely perform I/O. There will always be a need for some locking synchronization in an operating system, but operating systems should be able to take advantage of TM wherever possible. Given that transactions and locks will have to coexist in any realistic implementation, cooperation between locks and transactions is essential.

4. COOPERATION BETWEEN LOCKS AND TRANSACTIONS

In order to allow both transactions and conventional locks in the operating system, we propose a synchronization API that affords their seamless integration, called cooperative transactional spinlocks, or *cxspinlocks*. *Cxspinlocks* allow different executions of a single critical section to be synchronized with either locks or transactions. This freedom enables the concurrency of transactions when possible and enforces the safety of locks when necessary. Locking may be used for I/O, for protection of data structures read by hardware (e.g., the page table), or for high-contention access paths to particular data structures (where the performance of transactions might suffer from excessive restarts). The *cxspinlock* API also provides a simple upgrade path to let the kernel use transactions in place of existing synchronization.

Cxspinlocks are necessary for the kernel only; they allow the user programming model to remain simple. Users do not need them because they cannot directly access I/O devices (in Linux and most operating systems, users perform I/O by calling the OS). Blocking direct user access to devices is a common OS design decision that allows the OS to safely multiplex devices among noncooperative user programs. Sophisticated user programs that want transactions and locks to coexist can use *cxspinlocks*, but it is not required.

Using conventional Linux spinlocks within transactions is possible and will maintain mutual exclusion. However, conventional spinlocks reduce the concurrency of transactions and lacks fairness. Conventional spinlocks prevent multiple transactional threads from executing a critical region concurrently. All transactional threads in a critical region must read the spinlock memory location to obtain the lock and must write it to obtain the lock and release it. This write sharing among transactional threads will prevent concurrent execution, even if concurrent execution of the “real work” in the critical section is safe. Moreover, conventional spinlocks do not help with the I/O problem. A transactional thread that acquires a spinlock can restart, therefore it cannot perform I/O.

The progress of transactional threads can be unfairly throttled by non-transactional threads using spinlocks. In MetaTM conflicts between transactional and non-transactional threads (asymmetric conflicts) are always resolved in favor of the non-transactional thread. To provide isolation, HTM systems guarantee either that non-transactional threads always win asymmetric conflicts (like MetaTM), or transactional threads always win asymmetric conflicts (like Log-TM¹⁴). With either convention, traditional spinlocks will cause unfairness between transactional and non-transactional threads.

4.1. Cooperative transactional spinlocks

Cxspinlocks allow a single critical region to be safely protected by either a lock or a transaction. A non-transactional thread can perform I/O inside a protected critical section without concern for undoing operations on a restart. Many transactional threads can simultaneously enter critical sections protecting the same shared data, improving performance. Simple return codes in MetaTM allow the choice between locks and transactions to be made dynamically, simplifying programmer reasoning. *Cxspinlocks* ensure a set of behaviors that allow both transactional and non-transactional code to correctly use the same critical section while maintaining fairness and high concurrency:

- Multiple transactional threads may enter a single critical section without conflicting on the lock variable. A non-transactional thread will exclude both transactional and other non-transactional threads from entering the critical section.
- Transactional threads poll the *cxspinlock* using the *xtest* instruction, which allows a thread to check the value of a lock variable without entering the lock variable into the transaction’s read-set, enabling the transaction to avoid restarting when the lock is released (another thread writes the lock variable). This is especially important for acquiring nested *cxspinlocks* where the thread will have done transactional work before the attempted acquire.
- Non-transactional threads acquire the *cxspinlock* using an instruction (*xcas*) that is arbitrated by the transactional contention manager. This enables fairness between locks and transactions because the contention manager can implement many kinds of policies favoring transactional threads, non-transactional threads, readers, writers, etc.

Figure 2 shows the API and implementation. *Cxspinlocks* are acquired using two functions: *cx_exclusive* and *cx_optimistic*. Both functions take a lock address as an argument.

cx_optimistic is a drop-in replacement for spinlocks and is safe for almost all locking done in the Linux kernel (the exceptions are a few low-level page table locks and locks whose ownership is passed between threads, such as that protecting the run queue). *cx_optimistic* optimistically attempts to protect a critical section using transactions. If a code path within the critical section protected by *cx_optimistic* requires mutual exclusion, then the transaction restarts and acquires the lock exclusively. The code in Figure 1, which can fail due to I/O with bare transactions, functions with *cxspinlocks*, taking advantage of optimism with transactions when the *dentry_iput* function does no I/O, and retrying with with exclusive access when it does.

Control paths that will always require mutual exclusion (e.g., those that always perform I/O) can be optimized with *cx_exclusive*. Other paths that access the same data structure may execute transactionally using *cx_optimistic*. Allowing different critical regions to synchronize with a mix of

Figure 2: The cxspinlock API and implementation. The `cx_optimistic` API attempts to execute a critical section by starting a transaction, and using `xtest` to spin until the lock is free. If the critical section attempts I/O, the hardware will retry the transaction, returning the `NEED_EXCL` flag from the `xbegin` instruction. This will result in a call to the `cx_exclusive` API, which waits until the lock is free, and acquires the lock using the `xcas` instruction to atomically compare and swap the lock variable, and which invokes the contention manager to arbitrate any conflicts on the lock. The `cx_end` API exits a critical section, either by ending the current transaction, or releasing the lock.

cx_optimistic <i>Use transactions, restart on I/O attempt</i>	cx_exclusive <i>Acquire a lock, with contention manager</i>	cx_end <i>Release a critical section</i>
<pre>void cx_optimistic(lock) { status = xbegin; if(status==NEED_EXCL) { xend; if(gettxid) xrestart(NEED_EXCL); else cx_exclusive(lock); return; } while(!xtest(lock,1)); }</pre>	<pre>void cx_exclusive(lock) { while(1) { while(*lock != 1); if(xcas(lock, 1, 0)) break; } }</pre>	<pre>void cx_end(lock) { if(xgettxid) xend; else *lock = 1; }</pre>

`cx_optimistic` and `cx_exclusive` assures the maximum concurrency while maintaining safety.

4.2. Converting Linux to TxLinux-CX

While the TxLinux-SS conversion of the kernel replaces spinlocks in selected subsystems with bare transactions, TxLinux-CX replaces all spinlocks with cxspinlocks. The API addresses the limitations of transactions in an OS context, which not only made it possible to convert more locks, but made it possible to do it much more quickly: in contrast to the six developer years required to create TxLinux-SS, TxLinux-CX required a single developer-month.

5. HTM AWARE SCHEDULING

This section describes how MetaTM allows the OS to communicate its scheduling priorities to the hardware conflict manager, so the TM hardware does not subvert OS scheduling priorities or policy.

5.1. Priority and policy inversion

Locks can invert OS scheduling priority, resulting in a higher-priority thread waiting for a lower-priority thread. Some OSes, like Solaris, have mechanisms to deal with priority inversion such as priority inheritance, where a waiting thread temporarily donates its priority to the thread holding the lock. Recent versions of RT (Real-Time) Linux implement priority inheritance as well. Priority inheritance is complicated, and while the technique can shorten the length of priority inversion, it cannot eliminate it. In addition, it requires conversion of busy-waiting primitives such as spinlocks into blocking primitives such as mutexes. Conversion to mutexes provides an upper bound on latency in the face of priority inversion, but it slows down response time overall and does not eliminate the problem.

Simple hardware contention management policies for TM can invert the OS scheduling priority. HTM researchers have focused on simple hardware contention management that is guaranteed free from deadlock and livelock, e.g., timestamp, where the oldest transaction wins.²⁰ The timestamp policy

does not deadlock or livelock because timestamps are not refreshed during transactional restarts—a transaction will eventually become the oldest in the system, and it will succeed. But if a process with higher OS scheduler priority can start a transaction after a process with lower priority starts one and those transactions conflict, the timestamp policy will allow the lower priority process to continue if a violation occurs, and the higher priority process will be forced to restart.

Locks and transactions can invert not only scheduling priority, but scheduling policy as well. OSes that support soft real-time processes, like Linux, allow real-time threads to synchronize with non-real-time threads. Such synchronization can cause *policy inversion* where a real-time thread waits for a non-real-time thread. Policy inversion is more serious than priority inversion. Real-time processes are not just regular processes with higher priority, the OS scheduler treats them differently (e.g., if a real-time process exists, it will always be scheduled before a non-real-time process). Just as with priority inversion, many contention management policies bring the policy inversion of locks into the domain of transactions. A contention manager that respects OS scheduling policy can largely eliminate policy inversion.

The contention manager of an HTM system can nearly eradicate policy and priority inversion. The contention manager is invoked when the write-set of one transaction has a non-empty intersection with the union of the read- and write-set of another transaction. If the contention manager resolves this conflict in favor of the thread with higher OS scheduling priority, then transactions will not experience priority inversion.

5.2. Contention management using OS priority

To eliminate priority and policy inversion, MetaTM provides an interface for the OS to communicate scheduling priority and policy to the hardware contention manager (a mechanism suggested by other researchers^{14,22}). MetaTM implements a novel contention management policy called *os_prio*. The *os_prio* policy is a hybrid of contention management policies. The first prefers the transaction with the

greatest scheduling value to the OS. Given the small number of scheduling priority values, ties in conflict priority will not be rare, so *os_prio* next employs *timestamp*. This hybrid contention management policy induces a total order on transactions and is therefore livelock-free.

A single register in the MetaTM architecture allows the OS to communicate its scheduling priorities to the contention manager. TxLinux encodes a process' dynamic scheduling priority and scheduling policy into a single integer called the *conflict priority*, which it writes to a privileged status register during the process of scheduling the process. The register can only be written by the OS so the user code cannot change the value. The value of the register does not change during a scheduling quantum. For instance, the scheduling policy might be encoded in the upper bits of the conflict priority and the scheduling priority in the lower bits. An 8-bit value is sufficient to record the policies and priority values of Linux processes. Upon detecting a conflict, the *os_prio* contention manager favors the transaction whose conflict priority value is the largest.

The *os_prio* policy is free from deadlock and livelock because the conflict priority is computed before the *xbegin* instruction is executed, and the OS never changes the conflict priority during the lifetime of the transaction (some designs allow transactions to continue for multiple scheduling quanta). When priorities are equal, *os_prio* defaults to *Size-Matters*, which defaults to *timestamp* when read-write set sizes are equal. Hence, the tuple (*conflict priority*, *size*, *age*) induces a total order, making the *os_prio* policy free of deadlock and livelock.

6. EVALUATION

Linux and TxLinux versions 2.6.16.1 run on the Simics machine simulator version 3.0.27. In the following experiments, Simics models an x86 SMP machine with 16 and 32 processors and an IPC of one instruction per cycle. The memory hierarchy uses per-processor split instruction and data caches (16KB with 4-way associativity, 64-byte cache lines, 1-cycle cache hit and 16-cycle miss penalties). The level one data cache has extra tag bits to manage transactional data. There is a unified second level cache that is 4 MB, 8-way associative, with 64-byte cache lines, and a 200 cycle miss penalty to main memory. Coherence is maintained with a transactional MESI snooping protocol, and main memory is a single shared 1 GB. The disk device models PCI bandwidth limitations, DMA data transfer, and has a fixed 5.5 ms access latency. All benchmarks are scripted, requiring no user interaction.

6.1. Workloads

We evaluated TxLinux-SS and TxLinux-CX on a number of application benchmarks. Where *P* denotes the number of processors, these are:

- *pmake* (make -j *P* to compile part of libFLAC source tree)
- *MAB* (modified andrew benchmark, *P* instances, no compile phase)
- *configure* (configure script for a subset of TeTeX, *P* instances)

- *find* (Search 78MB directory (29 dirs, 968 files), *P* instances)
- *bonnie++* (models filesystem activity of a web cache, *P* instances)
- *dpunish* (a filesystem stress test, with operations split across *P* processes)

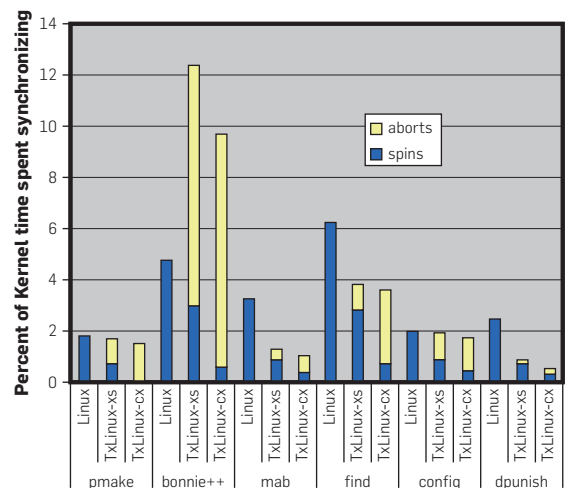
It is important to note that none of these benchmarks uses transactions directly; the benchmarks exercise the kernel, which in turn, uses transactions for synchronization.

6.2. Performance

Figure 3 shows the synchronization performance for Linux, TxLinux-SS (using bare transactions) and TxLinux-CX (using *cxspin*-locks) for 16 CPUs, broken down into time spent spinning on locks and time spent aborting and retrying transactions. Linux spends between 1% and 6% of kernel time synchronizing, while TxLinux spends between 1% and 12% of kernel time synchronizing. However, on average, TxLinux reduces synchronization time by 34% and 40% with transactions and *cxspin*locks, respectively. While HTM generally reduces synchronization overhead, it more than double the time lost for *bonnie++*. This loss is due to transactions that restart, back-off, but continue to fail. Since *bonnie++* does substantial creation and deletion of small files in a single directory, the resulting contention in file system code paths results in pathological restart behavior in the file system code that handles updates to directories. High contention on kernel data structures causes a situation in which repeated back-off and restart effectively starves a few transactions. Using back-off before restart as a technique to handle such high contention may be insufficient for complex systems: the transaction system may need to queue transactions that consistently do not complete.

Averaged over all benchmarks, TxLinux-SS shows a 2% slowdown over Linux for 16 CPUs and a 2% speedup for 32 CPUs. The slowdown in the 16 CPU case results from the pathological restart situation in the *bonnie++* benchmark, discussed above; the pathology is not present in the 32 CPU case with

Figure 3: Synchronization performance for 16 CPUs, TxLinux-SS, and TxLinux-CX.



bare transactions, and without *bonnie++*, TxLinux-SS shows the same speedup for 16 CPUs as 32 CPUs. TxLinux-CX sees 2.5% and 1% speedups over Linux for 16 and 32 CPUs. These performance deltas are negligible and do not demonstrate a conclusive performance increase. However, the argument for HTM in an operating system is about reducing programming complexity. These results show that HTM can enhance programmability without a negative impact on performance.

6.3. Priority inversion performance

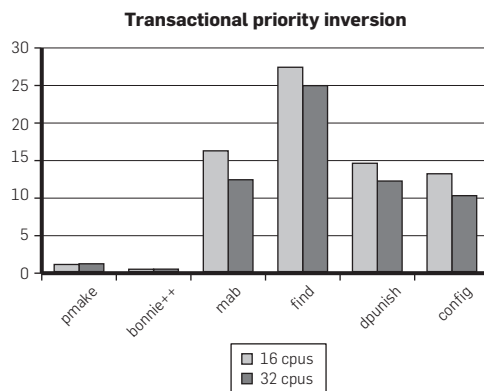
Figure 4 shows how frequently transactional priority inversion occurs in TxLinux. In this case, priority inversion means that the default *SizeMatters* contention management policy²¹ favors the process with the lower OS scheduling priority when deciding the winning transaction in a conflict. Results for timestamp-based contention management are similar. Most benchmarks show that a significant percentage of transactional conflicts result in a priority inversion, with the average 9.5% across all kernel and CPU configurations we tested, with as has 25% for *find*. Priority inversion tends to decrease with larger numbers of processors, but the trend is not strict. The *pmake* and *bonnie++* benchmarks show an increase with higher processor count. The number and distribution of transactional conflicts is chaotic, so changing the number of processors can change the conflict behavior. The *os_prio* contention management policy eliminates priority inversion entirely in our benchmarks, at a performance cost under 1%. By contrast, techniques for ameliorating priority inversion with locks such as priority inheritance only provide an upper bound on priority inversion, and require taking the performance hit of turning polling locks into blocking locks.

The frequency with which naïve contention management violates OS scheduling priority argues strongly for a mechanism that lets the OS participate in contention management, e.g., by communicating hints to the hardware.

7. RELATED WORK

Due to limited space, we refer the interested reader to the complete discussions,^{21,23} and survey the related literature in

Figure 4: Percentage of transaction restarts decided in favor of a transaction started by the processor with lower process priority, resulting in “transactional” priority inversion. Results shown are for all benchmarks, for 16 and 32 processors, TxLinux-SS.



brief. Larus and Rajwar provide a thorough reference on TM research through the end of 2006.¹²

HTM. Herlihy and Moss¹⁰ gave one of the earliest designs for HTM; many proposals since have focused on architectural mechanisms to support HTM,^{5,8,13,14,25} and language-level support for HTM. Some proposals for TM virtualization (when transactions overflow hardware resources) involve the OS,^{2,5} but no proposals to date have allowed the OS itself to use transactions for synchronization. This paper, however, examines the systems issues that arise when using HTM in an OS and OS support for HTM. Rajwar and Goodman explored speculative¹⁹ and transactional²⁰ execution of critical sections. These mechanisms for falling back on locking when isolation is violated are similar to (but less general than) the *cxspinlock* technique of executing in a transactional context and reverting to locking when I/O is detected.

I/O in transactions. Proposals for I/O in transactions fall into three basic camps: give transactions an isolation escape hatch,^{15–17} delay the I/O until the transaction commits,^{8,9} and guarantee that the thread performing I/O will commit.^{2,8} All of these strategies have serious drawbacks.¹¹ Escape hatches introduce complexity and correctness conditions that restrict the programming model and are easy to violate in common programming idioms. Delaying I/O is not possible when the code performing the I/O depends on its result, e.g., a device register read might return a status word that the OS must interpret in order to finish the transaction. Finally, guaranteeing that a transaction will commit severely limits scheduler flexibility, and can, for long-running or highly contended transactions, result in serial bottlenecks or deadlock. Non-transactional threads on other processors which conflict the guaranteed thread will be forced to stall until the guaranteed thread commits its work. This will likely lead to lost timer interrupts and deadlock in the kernel.

Scheduling. Operating systems such as Microsoft Windows, Linux, and Solaris implement sophisticated, priority-based, pre-emptive schedulers that provide different classes of priorities and a variety of scheduling techniques for each class. The Linux RT patch supports priority inheritance to help mitigate the effects of priority inversion: while our work also addresses priority inversion, the Linux RT patch implementation converts spinlocks to mutexes. While these mechanisms guarantee an upper bound on priority inversion, the *os_prio* policy allows the contention manager to effectively eliminate priority inversion without requiring the primitive to block or involve the scheduler.

8. CONCLUSION

This paper is the first description of an operating system that uses HTM as a synchronization primitive, and presents innovative techniques for HTM-aware scheduling and cooperation between locks and transactions. TxLinux demonstrates that HTM provides comparable performance to locks, and can simplify code while coexisting with other synchronization primitives in a modern OS. The *cxspinlock* primitive enables a solution to the long-standing problem of I/O in transactions, and the API eases conversion from locking primitives to transactions significantly. Introduction of

transactions as a synchronization primitive in the OS reduces time wasted synchronizing on average, but can cause pathologies that do not occur with traditional locks under very high contention or when critical sections are large enough to incur the overhead of HTM virtualization. HTM aware scheduling eliminates priority inversion for all the workloads we investigate. C

References

1. Adl-Tabatabai, A.-R., Lewis, B. T., Menon, V., Murphy, B. R., Saha, B., and Shpeisman, T. Compiler and runtime support for efficient software transactional memory. In *PLDI*, June 2006.
2. Blundell, C., Devietti, J., Lewis, E. C., and Martin, M. M. K. Making the fast case common and the uncommon case simple in unbounded transactional memory. In *ISCA*, 2007.
3. Carlstrom, B., McDonald, A., Chafi, H., Chung, J., Cao Minh, C., Kozyrakis, C., and Olukotun, K. The Atomos transactional programming language. In *PLDI*, June 2006.
4. Chou, A., Yang, J., Chelf, B., Hallem, S., and Engler, D. An empirical study of operating systems errors. In *SOSP*, 2001.
5. Chuang, W., Narayanasamy, S., Venkatesh, G., Sampson, J., Biesbrouck, M. V., Pokam, G., Calder, B., and Colavin, D. Unbounded page-based transactional memory. In *ASPLOS-XII*, 2006.
6. Elnozahy, E., Johnson, D., and Wang, Y. A survey of rollback-recovery protocols in message-passing systems, 1996.
7. Engler, D. and Ashcraft, K. Racer-X: Effective, static detection of race conditions and deadlocks. In *SOSP*, 2003.
8. Hammond, L., Wong, V., Chen, M., Carlstrom, B. D., Davis, J. D., Hertzberg, B., Prabhu, M. K., Wijaya, H., Kozyrakis, C., and Olukotun, K. Transactional memory coherence and consistency. In *ISCA*, June 2004.
9. Harris, T. Exceptions and side-effects in atomic blocks. *Sci. Comput. Program.*, 58(3):325–343, 2005.
10. Herlihy, M. and Moss, J. E. Transactional memory: Architectural support for lock-free data structures. In *ISCA*, May 1993.
11. Hofmann, O. S., Porter, D. E., Rossbach, C. J., Ramadan, H. E., and Witchel, E. Solving difficult HTM problems without difficult hardware. In *ACM TRANSACT Workshop*, 2007.
12. Larus, J. R. and Rajwar, R. *Transactional Memory*. Morgan & Claypool, 2006.

13. McDonald, A., Chung, J., Carlstrom, B., Minh, C. C., Chafi, H., Kozyrakis, C., and Olukotun, K. Architectural semantics for practical transactional memory. In *ISCA*, June 2006.
14. Moore, K. E., Bobba, J., Moravan, M. J., Hill, M. D., and Wood, D. A. Logtm: Log-based transactional memory. In *HPCA*, 2006.
15. Moravan, M. J., Bobba, J., Moore, K. E., Yen, L., Hill, M. D., Liblit, B., Swift, M. M., and Wood, D. A. Supporting nested transactional memory in logtm. In *ASPLOS-XII*, 2006.
16. Moss, E. and Hosking, T. Nested transactional memory: Model and preliminary architecture sketches. In *SCOOOL*, 2005.
17. Moss, J. E. B., Griffith, N. D., and Graham, M. H. Abstraction in recovery management. *SIGMOD Rec.*, 15(2):72–83, 1986.
18. Porter, D. E., Hofmann, O. S., and Witchel, E. Is the optimism in optimistic concurrency warranted? In *HotOS*, 2007.
19. Rajwar, R. and Goodman, J. Speculative lock elision: Enabling highly concurrent multithreaded execution. In *MICRO*, 2001.
20. Rajwar, R. and Goodman, J. Transactional lock-free execution of lock-based programs. In *ASPLOS*, 2002.
21. Ramadan, H., Rossbach, C., Porter, D., Hofmann, O., Bhandari, A., and Witchel, E. MetaTM/TxLinux: Transactional Memory for an Operating System. Evaluating transactional memory tradeoffs with TxLinux. In *ISCA*, 2007.
22. Ramadan, H., Rossbach, C., and Witchel, E. The Linux kernel: A challenging workload for transactional memory. In *Workshop on Transactional Memory Workloads*, June 2006.
23. Rossbach, C. J., Hofmann, O. S., Porter, D. E., Ramadan, H. E., Aditya, B., and Witchel, E. Txlinux: using and managing hardware transactional memory in an operating system. In *SOSP*, 2007.
24. Scherer III, W. N. and Scott, M. L. Advanced contention management for dynamic software transactional memory. In *PODC*, 2005.
25. Yen, L., Bobba, J., Marty, M., Moore, K. E., Volos, H., Hill, M. D., Swift, M. M., and Wood, D. A. Logtm-SE: Decoupling hardware transactional memory from caches. In *HPCA*, Feb 2007.

Christopher J. Rossbach, Hany E. Ramadan, Owen S. Hofmann, Donald E. Porter, Aditya Bhandari, and Emmett Witchel (rossbach,ramadan,osh, porterde,bhandari,witchel)@cs.utexas.edu, Department of Computer Sciences, University of Texas, Austin

© 2008 ACM 0001-0782/08/0900 \$5.00

ACM Transactions on Internet Technology



This quarterly publication encompasses many disciplines in computing—including computer software engineering, middleware, database management, security, knowledge discovery and data mining, networking and distributed systems, communications, and performance and scalability—all under one roof. TOIT brings a sharper focus on the results and roles of the individual disciplines and the relationship among them. Extensive multi-disciplinary coverage is placed on the new application technologies, social issues, and public policies shaping Internet development.

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

Technical Perspective

Distributing Your Data and Having It, Too

By Hagit Attiya

INTERCONNECTED SYSTEMS—the Internet, wireless networks, or sensor nets—embrace virtually all computing environments. Thus our data no longer needs to be stored, nicely organized, in centralized databases; it may instead span a great many heterogeneous locations and be connected through communication links. Records about stock-exchange transactions, for example, reside in broker firms and other financial entities around the world.

But data is worthless if it cannot be efficiently processed to yield useful information. Enter a data set's basic aggregates, such as the maximum, mean, or median, which can be combined to evaluate statistical properties of the data and guide decisions and actions, for example, by detecting trends stock markets.

Some of these aggregates can be computed quickly by means of a common network infrastructure—namely, a “spanning tree”—that connects all nodes storing information. For example, a simple recursive algorithm can compute the mean: each node averages the means computed in each subtree rooted at a node; and the resulting mean, together with the count, is then forwarded to the parent, which in turn averages the results from its own subtrees.

Assuming a unit of time that allows for receiving messages from all children and processing them; each iteration takes a single time unit, and the number of time units is proportional to the depth of the spanning tree, or diameter of the network, denoted D .

Essentially, the same algorithm computes the maximal or minimal value and similar aggregates. But the mean is sensitive to outliers: when the data has large variations the mean may misrepresent the data. On days when stock rates are highly fluctuating, a single stock transaction at an extremely low quote can significantly sway the mean, rendering it irrelevant. The me-

dian, other quartiles, or in general the k^{th} element of the data set are more significant in these situations.

But although the simple algorithm described here is fine for the mean, it does not work for computing the median, given that the median at an interior node is not necessarily the median of the medians in its sub-trees. A simple divide-and-conquer approach can be used instead. This algorithm starts with the entire set of elements and in each round randomly chooses a single *pivot* element. The algorithm then counts the number of elements larger than the pivot, and it recurses in the corresponding subinterval. A fairly straightforward analysis shows that when the pivots are chosen uniformly at random and the element counts are exact, then the expected number of iterations is asymptotically bounded by the logarithm of m , the number of nodes in the tree.

Putting this sequential algorithm to work in a large heterogeneous network, as done by Kuhn, Locher, and Wattenhofer, illustrates the challenges facing designers of network algorithms today and the innovations they must come up with to tackle them.


The main barrier is in sampling the pivot from a vast and scattered data set. The three researchers sidestep this barrier, however, by instead sending a search expedition to look for the pivot within the data. The search takes a random walk down the parent spanning tree, starting from the root and randomly choosing to which child to proceed. The choice is biased by the size of the sub-trees rooted at the children, but a careful tuning of the biases allows a pivot to be picked uniformly at random within time $2D$.

Instead of choosing one pivot at a time, the algorithm of Kuhn, Locher, and Wattenhofer finds D pivots within time $O(D)$ by staggering the search for several pivots in overlapping time intervals. Using D pivots, the number of candidates reduces by a factor of D in

each iteration, leading to a total time complexity of $O(D \log_b n)$, with n being the number of nodes in the network. Their paper then shows how to avoid the randomized pivot selection so as to make the algorithm deterministic, though at some cost.

Armed with these algorithms, the problem of computing data aggregates distributively is now largely solved. Data can be spread across a network, yet we can mine it to deduce important information.

And in case you were wondering: the answer is negative. No other algorithm can do better. The authors show that any algorithm for finding the median (and in general the k^{th} item) must take time proportional to $D \log_b n$, even if it can flip coins. In the true tradition of the theory of distributed computing, the lower bound follows from the uncertainty inherent in the problem.

The idea is to construct many scenarios, each with a different median, and use an information-theoretic argument to show that a lot of time is needed in order to disambiguate among these scenarios. 

Hagit Attiya (hagit@cs.technion.ac.il) is a professor of computer science at the Technion—Israel Institute of Technology, based in Haifa.

Distributed Selection: A Missing Piece of Data Aggregation

By Fabian Kuhn, Thomas Locher, and Roger Wattenhofer

Abstract

In this article, we study the problem of distributed selection from a theoretical point of view. Given a general connected graph of diameter D consisting of n nodes in which each node holds a numeric element, the goal of a k -selection algorithm is to determine the k^{th} smallest of these elements. We prove that distributed selection indeed requires more work than other aggregation functions such as, e.g., the computation of the average or the maximum of all elements. On the other hand, we show that the k^{th} smallest element can be computed efficiently by providing both a randomized and a deterministic k -selection algorithm, dispelling the misconception that solving distributed selection through in-network aggregation is infeasible.

1. INTRODUCTION

There is a recent growing interest in distributed aggregation, thanks to emerging application areas such as, e.g., data mining or sensor networks.^{2,8,23,24} The goal of distributed aggregation is to compute an aggregation function on a set of distributed values, each value stored at a node in a network. Typical aggregation functions are *max*, *sum*, *count*, *average*, *median*, *variance*, k^{th} *smallest*, or *largest value*, or combinations thereof such as, e.g., “What is the average of the 10% largest values?”

The database community classifies aggregation functions into three categories: distributive (*max*, *min*, *sum*, *count*), algebraic (*plus*, *minus*, *average*, *variance*), and holistic (*median*, k^{th} *smallest*, or *largest value*). Combinations of these functions are believed to support a wide range of reasonable aggregation queries.*

It is well known that distributive and algebraic functions can easily be computed using the so-called *convergecast* operation executed on a pre-computed *breadth first search* (BFS) tree: The root of the tree floods a message to the leaves of the tree, asking the leaves to start the aggregation. The inner nodes of the spanning tree wait until they have received the aggregated data from all their children, apply the aggregation function to their own data and the aggregated data, and subsequently forward the aggregation result to their respective parent. Convergecast is fast, as it terminates after at most $2D_T$ time, where D_T denotes the depth of the spanning tree. Note that the depth of a BFS tree is at most the diameter D of the original graph G , thus a single convergecast costs merely $2D$ time. An example for such a spanning tree in the context of sensor networks is depicted in Figure 1. However,

*We encourage the reader to think of a natural aggregation (single value result) query that cannot be formulated by a combination of distributive, algebraic, and holistic functions.

it is believed that holistic functions cannot be supported by convergecast. After all, the very name “holistic” indicates that one “cannot look into” the set of values, more precisely, that all the values need to be centralized at one node in order to compute the holistic function. Bluntly, in-network aggregation is considered to be practically impossible for holistic functions.

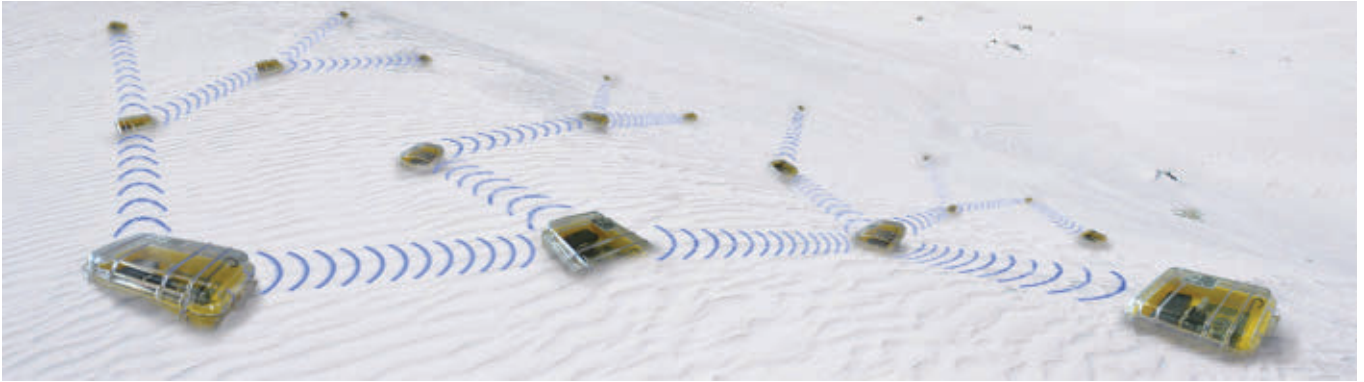
For arbitrary k , a selection algorithm answers questions about the k^{th} smallest value in a set or network. The special case of the k -selection problem where $k = n/2$ is the well-known *median problem*. Generally speaking, selection solves aggregation queries about order statistics and percentiles. Surprisingly, little is known about distributed (network) selection, although it is critical to the understanding of data aggregation.

In this article, we shed some new light on the problem of distributed selection for general networks with n nodes and diameter D . In particular, we prove that distributed selection is strictly harder than convergecast by giving a lower bound of $\Omega(D \log_p n)$ on the time complexity in Section 5. In other words, to the best of our knowledge, we are the first to formally confirm the preconception about holistic functions being strictly more difficult than distributive or algebraic functions. In addition, in Section 4.1, we present a novel *Las Vegas algorithm* which matches this lower bound with high probability, improving the best randomized algorithm. As for many networks this running time is strictly below collecting all values at one node, our new upper bound proves that (contrary to common belief) in-network aggregation is possible also for holistic functions; in fact, in network topologies where the diameter is large, e.g., in grids or in typical wireless sensor networks, selection can be performed within the same asymptotic time bounds as convergecast. As a third result, in Section 4.2, we derandomize our algorithm and arrive at a deterministic distributed selection algorithm with a time complexity of $O(D \log_D^2 n)$ which constitutes a substantial improvement over prior art.

2. RELATED WORK

Finding the k^{th} smallest value among a set of n elements is a classic problem which has been extensively studied in the past approximately 30 years, both in distributed and non-distributed settings. The problem of finding the median, i.e., the element for which half of all elements are smaller and the other half is larger, is a special case of the k -selection problem which has also received a lot of attention. Blum et al.¹ proposed the first deterministic sequential algorithm that, given an array of size n , computes the k^{th} smallest element in $O(n)$ time. The algorithm partitions the n elements into roughly $n/5$ groups of five elements and determines the

Figure 1: Data in sensor networks is aggregated on a pre-computed virtual spanning tree. Typically, nodes receive data from their children and forward the aggregation result to their parent in the tree.



median element of each group. The median of these $n/5$ medians is then computed recursively. While this median of medians is not necessarily the median among all n elements, it still partitions all elements well enough in that at least (roughly) 30% of all elements are smaller, and also at least 30% are larger. Thus, at least 30% of all elements can be excluded and the algorithm can be applied recursively to the remaining elements. A careful analysis of this algorithm reveals that only $O(n)$ operations are required in total. Subsequently, Schönhage et al.¹⁹ developed an algorithm requiring fewer comparisons in the worst-case.

As far as distributed k -selection is concerned, a rich collection of algorithms has been amassed for various models over the years. A lot of work focused on special graphs such as stars and complete graphs.^{9,16} The small graph consisting of two connected nodes where each node knows half of all n elements has also been studied and algorithms with a time complexity of $O(\log n)$ have been presented.^{3,15} For deterministic algorithms in a restricted model, this result has been shown to be tight.¹⁵ Frederickson¹⁴ proposed algorithms for rings, meshes, and also complete binary trees whose time complexities are $O(n)$, $O(\sqrt{n})$, and $O(\log^3 n)$, respectively.

Several algorithms, both of deterministic^{12,13,20} and probabilistic nature,^{17,18,20} have also been devised for arbitrary connected graphs. Some of these deterministic algorithms restrict the elements the nodes can hold in that the maximum numeric item x_{max} has to be bounded by $O(n^{O(1)})$. Given this constraint, applying binary search results in a time complexity of $O(D \log x_{max}) = O(D \log n)$.¹³ Alternatively, by exponentially increasing the initial guess of $x_k = 1$, the solution can be found in $O(D \log x_k)$.¹² To the best of our knowledge, the only non-restrictive deterministic k -selection algorithm for general graphs with a sublinear time complexity in the number of nodes is due to Shrira et al.²⁰ Their adaptation of the classic sequential algorithm by Blum et al. for a distributed setting has a worst-case running time of $O(Dn^{0.9114})$. In the same work, a randomized algorithm for general graphs is presented. The algorithm simply inquires a random node for its element and uses this guess to narrow down the number of potential elements. The expected time complexity is shown to be $O(D \log n)$. Kempe et al.⁷ proposed a gossip-based algorithm that, with probability at least $1 - \epsilon$, computes the k^{th}

smallest element within $O((\log n + \log \frac{1}{\epsilon}) + (\log n + \log \log \frac{1}{\epsilon}))$ rounds of communication on a complete graph.

If the number of elements N is much larger than the number of nodes, in $O(D \log \log \min\{k, N - k + 1\})$ expected time, the problem can be reduced to the problem where each node has exactly one element using the algorithm proposed by Santoro et al.^{17,18} However, their algorithm depends on a particular distribution of the elements on the nodes. Patt-Shamir¹³ showed that the median can be approximated very efficiently, again subject to the constraint that the maximum element must be bounded by a polynomial in n .

3. MODEL AND DEFINITIONS

In our system model, we are given a connected graph $G = (V, E)$ of diameter D with node set V and edge set E . The cardinality of the node set is $|V| = n$ and the nodes are denoted v_1, \dots, v_n . The *diameter* of a graph is the length of the longest shortest path between any two nodes. Each node v_i holds a single element x_i .[†] Without loss of generality, we can assume that all elements x_i are unique. If two elements x_i and x_j were equal, node IDs, e.g., i and j , could be used as tiebreakers. The goal is to efficiently compute the k^{th} smallest element among all elements x_1, \dots, x_n , and the nodes can achieve this goal by exchanging messages. Nodes v_i and v_j can directly communicate if $(v_i, v_j) \in E$.

The standard asynchronous model of communication is used. Throughout this article, the communication is considered to be reliable, there is no node failure, and all nodes obediently follow the mandated protocol. We do not impose any constraint on the magnitude of the stored elements. However, we restrict the size of any single message such that it can contain solely a constant number of both node IDs and elements, and also at most $O(\log n)$ arbitrary additional bits. By restricting the size of the messages, we strive to capture how much *information* has to be exchanged between the nodes in order to solve the problem. Moreover, such a restriction is quite natural as the message size is typically

[†] Our results can easily be generalized to the case where more than one element is stored at each node. The time complexities are then stated in terms of the number of elements $N > n$ instead of the number of nodes.

limited in practical applications. Note that without this restriction on the message size, a single convergecast would suffice to accumulate all elements at a single node, which could subsequently solve the problem locally.

As both proposed algorithms are *iterative* in that they continuously reduce the set of possible solutions, we need to distinguish between nodes holding elements that are still of interest from the other nodes. Henceforth, the first set of nodes is referred to as *candidate nodes* or *candidates*. We call the reduction of the search space by a certain factor a *phase* of the algorithm. The number of candidate nodes in phase i is denoted $n^{(i)}$.

We assume that all nodes know the diameter D of the graph. Furthermore, it is assumed that a BFS spanning tree rooted at the node initiating the algorithm has been computed beforehand. These assumptions are not critical as both the diameter and the spanning tree can be computed in $2D$ time.¹⁴

The main complexity measure used is the time complexity which is, for deterministic algorithms, the time required from the start of an execution to its completion in the worst-case for every legal input and every execution scenario. The time complexity is normalized in that the slowest message is assumed to reach its target after one time unit. As far as our randomized algorithm is concerned, we determine the time after which the execution of the algorithm has completed with high probability, i.e., with probability at least $1 - \frac{1}{c}$ for a constant $c \geq 1$. Thus, in both cases, we do not assign any cost to local computation.

4. ALGORITHMS

The algorithms presented in this article operate in sequential phases in which the space of candidates is steadily reduced. This pattern is quite natural for k -selection and used in all other proposed algorithms including the non-distributed case. The best known deterministic distributed algorithm for general graphs uses a distributed version of the well-known *median-of-median* technique, which we briefly outlined in Section 2, resulting in a time complexity of $O(Dn^{0.9114})$ for a constant group size. A straightforward modification of this algorithm in which the group size in each phase i is set to $O(\sqrt{n^{(i)}})$ results in a much better time complexity. It can be shown that the time complexity of this variant of the algorithm is bounded by $O(D(\log n)^{\log n + O(1)})$. However, since our proposed algorithm is substantially better, we will not further discuss this median-of-median-based algorithm. Due to the more complex nature of the deterministic algorithm, we will treat the randomized algorithm first.

4.1 Randomized algorithm

While the derivation of an expedient deterministic algorithm is somewhat intricate, it is remarkably simple to come up with a fast randomized algorithm. An apparent solution, proposed by Shrira et al.,²⁰ is to choose a node randomly and take its element as an initial guess. After computing the number of nodes with smaller and larger elements, it is likely that a considerable fraction of all nodes no longer need be considered. By iterating this procedure on the

remaining candidate nodes, the k^{th} smallest element can be found quickly for all k .

A node can be chosen randomly using the following scheme: A message indicating that a random element is to be selected is sent along a random path in the spanning tree starting at the root. If the root has l children v_1, \dots, v_l where child v_i is the root of a subtree with n_i candidate nodes including itself, the root chooses its own element with probability $1/(1 + \sum_{j=1}^l n_j)$. Otherwise, it sends a message to one of its children. The message is forwarded to node v_i with probability $n_i/(1 + \sum_{j=1}^l n_j)$ for all $i \in \{1, \dots, l\}$, and the recipient of the message proceeds in the same manner. It is easy to see that this scheme selects a node uniformly at random and that it requires at most $2D$ time, because the times to reach any node and to report back are both bounded by D . Note that after each phase the probabilities change as they depend on the altered number of candidate nodes remaining in each subtree. However, having determined the new interval in which the solution must lie, the number of nodes satisfying the new predicate in all subtrees can again be computed in $2D$ time.

This straightforward procedure yields an algorithm that finds the k^{th} smallest element in $O(D \log n)$ expected time, as $O(\log n)$ phases suffice in expectation to narrow down the number of candidates to a small constant. It can even be shown that the time required is $O(D \log n)$ with high probability. The key observation to improve this algorithm is that picking a node randomly always takes $O(D)$ time, therefore several random elements ought to be chosen in a single phase in order to further reduce the number of candidate nodes. The method to select a single random element can easily be modified to allow for the selection of several random elements by including the number of needed random elements in the request message. A node receiving such a message locally determines whether its own element is chosen, and also how many random elements each of its children's subtrees has to provide. Subsequently, it forwards the requests to all of its children whose subtrees must produce at least one random element. Note that all random elements can be found in D time independent of the number of random elements, but due to the restriction that only a constant number of elements can be packed into a single message, it is likely that not all elements can propagate back to the root in D time. However, all elements still arrive at the root in $O(D)$ time if the number of random elements is bounded by $O(D)$.

By using this simple *pipelining* technique to select $O(D)$ random elements in $O(D)$ time, we immediately get a more efficient algorithm, which we will henceforth refer to as A^{rand} . When selecting $O(D)$ elements uniformly at random in each phase, it can be shown that the number of candidates is reduced by a factor of $O(D)$ in a constant number of phases with high probability with respect to $O(D)$, as opposed to merely a constant factor in case only a single element is chosen. This result, together with the observation that each phase costs merely $O(D)$ time, is used to prove the following theorem.

THEOREM 4.1. *In a connected graph of diameter $D \geq 2$ consisting of n nodes, the time complexity of algorithm A^{rand} is $O(D \log_p n)$ w.h.p.*

In particular in graphs where D is large, algorithm A^{rand} is considerably faster than the algorithm selecting only a single random element in each phase. In Section 5, we prove that no deterministic or probabilistic algorithm can be better asymptotically, i.e., A^{rand} is *asymptotically optimal*.

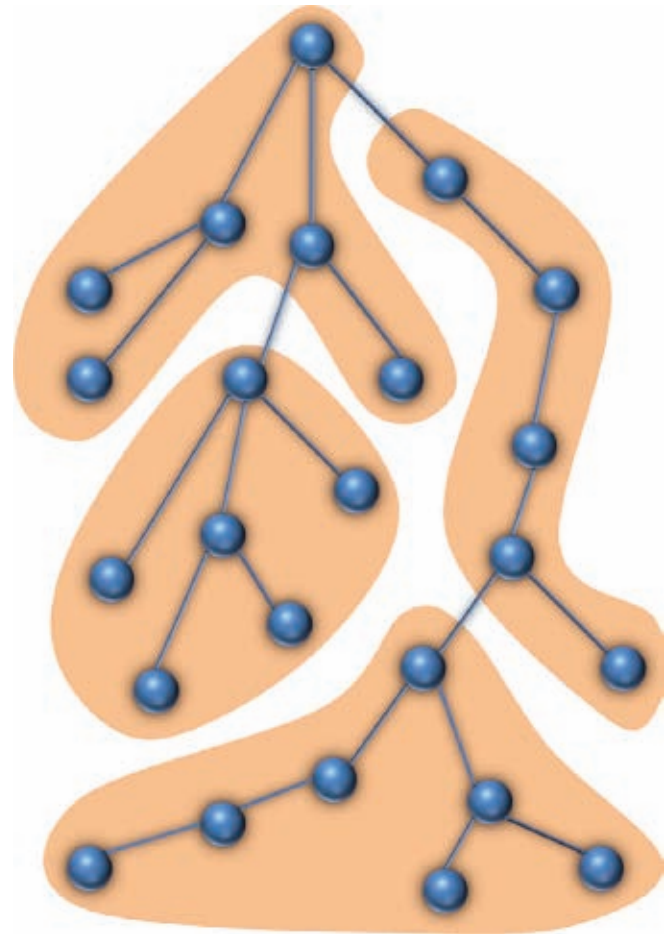
4.2 Deterministic algorithm

The difficulty of deterministic iterative algorithms for k -selection lies in the selection of elements that provably allow for a reduction of the search space in each phase. Once these elements have been found, the reduced set of candidate nodes can be determined in the same way as in the randomized algorithm. Thus, the deterministic algorithm, referred to as A^{det} , has to compute a set of $O(D)$ elements that partitions all elements similarly to the random set used by algorithm A^{rand} in each phase.

A simple idea to go about this problem is to start sending up elements from the leaves of the spanning tree, accumulating the elements from all children at the inner nodes, and then recursively forwarding a selection of t elements to the parent. The problem with this approach is the reduction of all elements received from the children to the desired t elements. If a node v_i receives t elements from each of its c_i children in the spanning tree, the t elements that partition all $c_i t$ nodes into segments of approximately equal size ought to be found. However, in order to find these elements, the number of elements in each segment has to be counted starting at the leaves. Since this counting has to be repeated in each step along the path to the root, the time required to find a useful partitioning into k segments requires $O(D(D + Ct))$ time, where $C := \max_{i \in \{1, \dots, n\}} c_i$. This approach suffers from several drawbacks: It takes at least $O(D^2)$ time just to find a partitioning, and the time complexity depends on the structure of the spanning tree.

Our proposed algorithm A^{det} solves these issues in the following manner. In any phase i , the algorithm splits the entire spanning tree into $O(\sqrt{D})$ groups, each of size $O(n^{(i)}/\sqrt{D})$. Figure 2 depicts an example tree split into 4 groups. Recursively, in each of those groups a particular node initiates the same partitioning into $O(\sqrt{D})$ groups as long as the group size is larger than $O(\sqrt{D})$. The goal of this recursive partitioning is to find, for each group, $O(\sqrt{D})$ elements that reduce the search space by a factor of $O(\sqrt{D})$. Groups of size at most $O(\sqrt{D})$ can simply report all their elements to the node that initiated the grouping at this recursion level. Once such an initiating node v has received all $O(\sqrt{D})$ elements from each of the $O(\sqrt{D})$ groups it created, it sorts those $O(D)$ elements, and subsequently issues a request to count the nodes in each of the $O(D)$ intervals induced by the received elements. Assume that all the groups created by node v together contain $n_v^{(i)}$ nodes in phase i . The intervals can locally be merged into $O(\sqrt{D})$ intervals such that each interval contains at most $O(n_v^{(i)}/\sqrt{D})$ nodes. These $O(\sqrt{D})$ elements are recursively sent back to the node that created the group to which node v belongs. Upon receiving the $O(D)$ elements from its $O(\sqrt{D})$ groups and counting the number of nodes in each interval, the root can initiate phase $i + 1$ for which it holds that $n^{(i+1)} < n^{(i)}/O(\sqrt{D})$.

Figure 2: An example tree of diameter 12 consisting of 24 nodes is split according to algorithm A^{det} into 4 groups, each consisting of at most 7 nodes.



Given that the number of candidates reduces by a factor of $O(\sqrt{D})$ in each phase, it follows that the number of phases is bounded by $O(\log_D n)$. It can be shown that each phase costs $O(D \log_D n)$ time, which proves the following bound on the time complexity.

THEOREM 4.2. *In a connected graph of diameter $D \geq 2$ consisting of n nodes, the time complexity of algorithm A^{det} is $O(D \log_D^2 n)$.*

5. LOWER BOUND

In this section, we sketch how to prove a time lower bound for *generic* distributed selection algorithms which shows that the time complexity of the simple randomized algorithm of Section 4.1 for finding the element of rank k is asymptotically optimal for most values of k . Informally, we call a selection algorithm generic if it does not exploit the structure of the element space except for using the fact that there is a global order on all the elements. Formally, this means that the only access to the structure of the element space is by means of the comparison function. Equivalently, we can assume that all elements assigned to the nodes are fixed but that the ordering of elements belonging to different nodes is determined by an adversary and is initially not known to the

nodes. For the lower bound, we use a simpler synchronous communication model where time is divided into rounds and in every round each node can send a message to each of its neighbors. Note that since the synchronous model is strictly more restrictive than the asynchronous model, a lower bound for the synchronous model directly carries over to the asynchronous model. We show that if in any round only one element can be transmitted over any edge, such an algorithm needs at least $\Omega(D \log_D n)$ rounds to find the median with reasonable probability.

Generalizing the lower bound to finding the element of rank k for arbitrary $k \in \{1, \dots, n\}$ is straight-forward. We can assume that $k \leq n/2$ because finding the element of rank k is equivalent to finding the element of rank $n + 1 - k$ with respect to the inverse global order. We can now just inform the algorithm about the rank of all but the first $2k$ elements (additional information cannot make the problem harder). The problem now reduces to finding the median of $2k$ elements.

Let us start with an outline of our proof strategy. The lower bound is proven in two steps. We first consider protocols between two nodes where each of the nodes starts with half of the elements. Assuming that the two nodes can send each other messages containing at most $B \geq 1$ elements in each round, we show that $\Omega(\log_B n)$ rounds are needed to find the median. In a second step, we obtain the desired lower bound for general graphs by means of a reduction: We construct a graph $G(D)$ for every diameter $D \geq 3$ such that every T -round median algorithm on $G(D)$ can be turned into a $T/(D - 2)$ -round two-party protocol in which the two nodes have to communicate $D - 2$ elements per message.

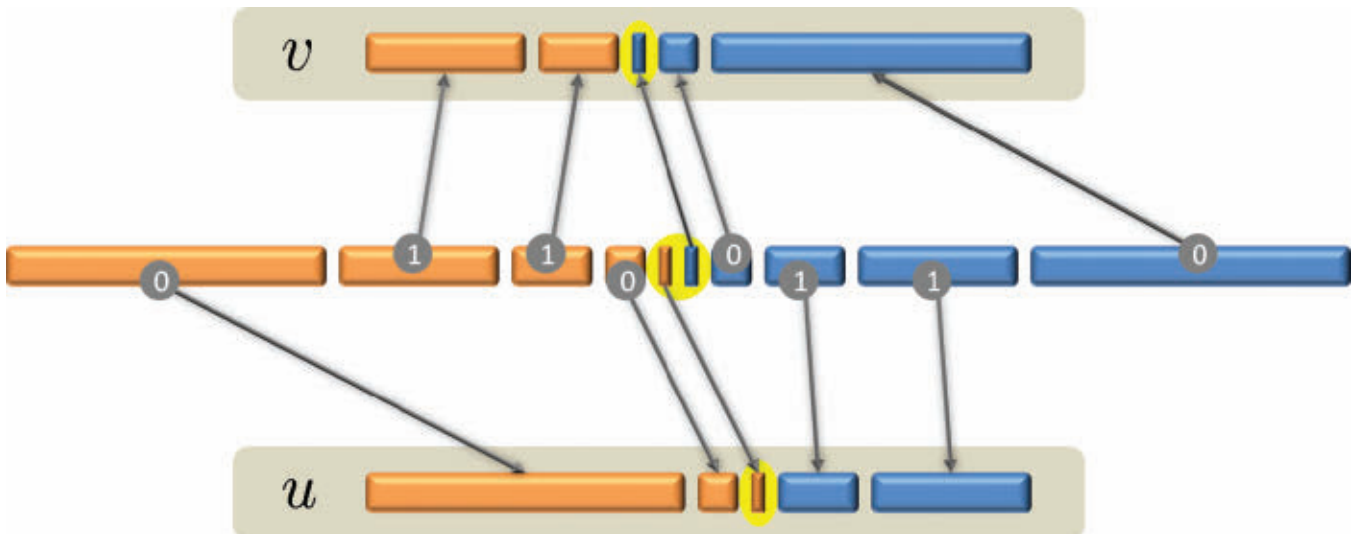
We therefore start by studying protocols between two nodes u and v such that u and v each have $N \geq 1$ elements $u_0 < u_1 < \dots < u_{N-1}$ and $v_0 < v_1 < \dots < v_{N-1}$ respectively, where $<$ is the global order according to which we want to find the median. We denote the sets of elements of u and v by S_u and S_v , respectively. Each message $M = (S, X)$ between the two nodes is further assumed to contain a set S of at

most B elements and some arbitrary additional information X . Assume M is a message from u to v . In this case, X can be everything which can be computed from the results of the comparisons between all the elements u has seen so far, as well as all the additional information u has received so far. The only restriction on X is that it cannot be used to transmit information about the values of elements not in S or in one of the earlier messages. We call a protocol between u and v which only sends messages of the form $M = (S, X)$ as described above, a generic two-party protocol.

The general idea is as follows. We define N different partitions, each assigning N of the $2N$ elements to u , and the other N elements to v (i.e., N different orders between the elements in S_u and S_v) in such a way that each partition results in a different median element. We choose as input one of the N partitions uniformly at random. In order to compute the median, we then have to find out which of the N partitions was chosen. We show that in each communication round, the probability for reducing the number of possible partitions by more than a factor of λB is exponentially small in λ .

For simplicity, assume that $N = 2^l$ is a power of 2. Let $X_0, \dots, X_{l-1} \sim \text{Bernoulli}(1/2)$ be l independent Bernoulli variables, i.e., all X_i take values 0 or 1 with equal probability. The partition of the $2N$ elements among u and v is determined by the values of X_0, \dots, X_{l-1} . If $X_{l-1} = 0$, the $N/2$ smallest of the $2N$ elements are assigned to u and the $N/2$ largest elements are assigned to v . If $X_{l-1} = 1$, it is the other way round. In the same way, the value of X_{l-2} determines the assignment of the smallest and largest $N/4$ of the remaining elements: If $X_{l-2} = 0$, u gets the elements with ranks $N/2 + 1, \dots, 3N/4$ and v gets the elements with ranks $5N/4 + 1, \dots, 3N/2$ among all $2N$ elements. Again, the remaining elements are recursively assigned analogously depending on the values of X_{l-3}, \dots, X_0 until only the two elements with ranks N and $N + 1$ (i.e., the two median elements) remain. The element with rank N is assigned to u and the element with rank $N + 1$ is assigned to v . Figure 3 illustrates the described process.

Figure 3: The $2N$ elements minus the two medians are assigned to u and v according to $l = \log n$ independent Bernoulli variables X_0, \dots, X_{l-1} . One of the two medians is assigned to u and the other to v . In order to find the medians, u and v must compute the values of X_0, \dots, X_{l-1} .



Consider the two elements u_{α^*} and v_{β^*} with ranks N and $N + 1$, respectively, and let α^* and β^* be the ranks of the elements N and $N + 1$ within the sets S_u and S_v . We consider the median problem to be solved as soon as either u knows α^* or v knows β^* . The elements are partitioned in such a way that the random variables X_i directly determine the base-2 representations of α^* and β^* . If $X_0 = 0$, the most significant bit of the bit representation of α^* is 1, whereas the most significant bit of the bit representation of β^* is 0. If $X_0 = 1$, the most significant bit of α^* is 0 and the most significant bit of β^* is 1. The other bits of the base-2 representations of α^* and β^* are determined analogously: If $X_i = 0$, the $(i + 1)$ st-most significant bit of α^* is 1 and the $(i + 1)$ st-most significant bit of β^* is 0 and vice versa if $X_i = 1$. Consider two arbitrary elements $u_\alpha \in S_u$ and $v_\beta \in S_v$ with ranks α and β within the two sets S_u and S_v , respectively. The outcome of the comparison of u_α and v_β (i.e., whether $u_\alpha < v_\beta$ or $v_\beta < u_\alpha$) is determined by the first variable X_i that is equal to the corresponding bit in the base-2 representation of u_α or different from the corresponding bit in the base-2 representation of v_β , whatever occurs first. If $X_i = 0$, we have that $u_\alpha < v_\beta$, otherwise $v_\beta < u_\alpha$.

Clearly, u and v can only learn about α^* and β^* from comparisons between their own elements and elements they have received from the other node and from additional information that the nodes sent to each other. Consider a generic two-party algorithm A that computes the median. Assume that after a certain time of the execution of A , $u_{\alpha_1}, \dots, u_{\alpha_r}$ are the elements that u has sent to v and $u_{\beta_1}, \dots, u_{\beta_s}$ are the elements that v has sent to u . Let \hat{i} be the largest index such that there is an element $u_{\beta_{\hat{i}}}$ for which the first \hat{i} bits are different from the corresponding variable X_i or such that there is an element $v_{\beta_{\hat{i}}}$ for which the first \hat{i} bits are equal to the corresponding variable X_i . By the above observation, any comparison between an element in S_u and an element that v has sent to u and any comparison between an element in S_v and an element that u has sent to v is determined by the values of $X_0, \dots, X_{\hat{i}-1}$. Intuitively, u and v cannot have any information about the values of $X_{\hat{i}}, \dots, X_{l-1}$. Thus, u and v have to guess the remaining bits by sending each other the right elements. It can be shown that the probability for guessing at least ξ bits correctly in a single round is at most $2B/2^{\xi}$. The number of newly learned bits in each round can be upper bounded by independent random variables. Using a Chernoff-type argument, one can then show that $\log_{2B}(N)/c$ rounds are needed to learn all l bits X_0, \dots, X_{l-1} with probability at least $1 - 1/N^{1-c}$, implying the following theorem.

THEOREM 5.1. *Every, possibly randomized, generic two-party protocol to find the median needs at least $\Omega(\log_{2B} N)$ rounds in expectation and with probability at least $1 - 1/N^\delta$ for every constant $\delta < 1/2$.*

Based on the lower bound for two-party protocols, we can now prove a lower bound for generic selection algorithms on general graphs. In the following, we assume that every node of a graph with n nodes starts with one element and that we have to find the k^{th} smallest of all n elements. In every round, every node can send one element to each of its neighbors.

For every $n \geq D \geq 3$, we construct a graph $G(D)$ with n nodes and diameter D such that we can reduce the problem of finding the median by a two-party protocol to the problem of finding the element of rank k in $G(D)$.

We first describe a lower bound of $\Omega(D \log_D n)$ for finding the median and then generalize to finding the element of an arbitrary rank k . For simplicity, assume that $n - D$ is an odd number. Let $N = (n - D + 1)/2$. We consider the graph $G(D)$ defined as follows: The graph $G(D)$ consists of two nodes u and v that are connected by a path of length $D - 2$ (i.e., it contains $D - 1$ nodes). In addition, there are nodes u_1, \dots, u_N and v_1, \dots, v_N such that u_i is connected to u and v_i is connected to v for all $i \in \{1, \dots, N\}$. We can certainly assume that $n = \omega(D)$ because $\Omega(D)$ is a trivial lower bound (even finding the minimum element requires $\Omega(D)$ rounds). We can therefore assume that only the leaf nodes u_i and v_i for $i \in \{1, \dots, N\}$ hold an element and that we need to find the median of these $2N$ elements. We can simply assign dummy elements to all other nodes such that the global median is equal to the median of the leaf elements. Since only the leaves start with an element, we can assume that in the first round, all leaves u_i send their element to u and all leaves v_i send their element to v , as this is the only possible useful communication in the first round. By this, the problem reduces to finding the k^{th} smallest element of $2N$ elements on a path of length $D - 2$ if initially each of the two end nodes u and v of the path holds N elements. Note that the leaf nodes of $G(D)$ do not need to further participate in a distributed selection protocol since u and v know everything their respective leaves know and can locally simulate all actions of their leaf nodes.

Assume that we are given an algorithm A that finds the median on $G(D)$ in time $T + 1$. We sketch how to construct a two-party protocol A' that sends at most $D - 2$ elements per message and finds the median in time $\lceil T/(D - 2) \rceil$. The $\Omega(\log_D N)$ lower bound for such an algorithm A' then implies that $T = \Omega(D \log_D N)$. Because information needs at least $D - 2$ to travel from u to v and vice versa, everything u and v can compute in round t (i.e., after receiving the message from round $t - 1$) is a function of their own elements and the content of the messages of the other node up to round $t - (D - 2)$. For example, u 's messages of the first $D - 2$ rounds only depend on S_u , the messages of rounds $D - 1, \dots, 2(D - 2)$ can be computed from S_u and v 's messages in the first $D - 2$ rounds, and so on. To obtain a two-party protocol A' from A , we can proceed as follows. In the first round of A' , u and v send their messages of the first $D - 2$ rounds of A to each other. Now, u and v can both locally simulate all communication of the first $D - 2$ rounds of A . This allows to compute the messages of the next $D - 2$ rounds of A . In general, in round r of the two-party protocol A' , u and v can send each other their messages of rounds $(r - 1)(D - 2) + 1, \dots, r(D - 2)$ of A and can afterwards locally simulate the respective rounds of A . The time complexity of A' then becomes $\lceil T/(D - 2) \rceil$.

THEOREM 5.2. *For every $n \geq D \geq 3$, there is a graph $G(D)$ with n nodes and diameter D such that every, possibly randomized, generic algorithm to find the k^{th} smallest element requires*

$\Omega(D \log_D \min\{k, n - k\})$ rounds in expectation and with probability at least $1 - 1/(\min\{k, n - k\})^\delta$ for every constant $\delta < 1/2$. In particular, finding the median requires at least $\Omega(D \log_D n)$ rounds.

Similar proof techniques have been used in the area of communication complexity where people try to find bounds on the total number of bits that have to be transmitted to solve a certain communication problem.²² In particular,²¹ introduces a simulation technique to run two-party protocols in paths and more general graphs in order to extend lower bound for computations between two nodes to computations on more general topologies. In contrast to communication complexity, our focus is on minimizing the number of communication rounds rather than minimizing the number of transmitted bits.

6. CONCLUSION

In this article, we studied the k -selection problem, a prominent data aggregation problem, and proved upper and lower bounds on its (time) complexity. Our results are presented in an entirely abstract way, i.e., it remains to show that our algorithms have a notable impact on the performance of aggregation functions in real networks and thus prove to be relevant in practical applications. Apparently, it is usually not possible to simply implant a distributed algorithm in an application domain, as additional constraints imposed by the application need to be respected. In wireless sensor networks, e.g., aggregation needs to adhere to wireless channel characteristics. We believe that our work can shed additional light on the achievable aggregation rate and capacity^{10,11} in wireless networks, or, combined with other algorithmic work on data gathering such as, e.g.,^{5,6} may even provide basic aggregation functionality for various application domains. We hope that our results and techniques may eventually find their way into several application areas, providing aggregation support for, e.g., streaming databases or multi-core architectures.

Acknowledgments

We would like to thank Pascal von Rickenbach and Roland Flury for their help with the illustrations. Moreover, we would like to express our gratitude to Hagit Attiya for providing valuable feedback, which helped us greatly to improve this article, and also for finding the time to write a technical perspective.

The original version of this paper is entitled “Tight Bounds for Distributed Selection” and can be found in the *Proceedings of the 19th ACM Symposium on Parallelism in Algorithms and Architectures* (San Diego, CA, June 2007). ■

References

- Blum, M., Floyd, R. W., Pratt, V., Rivest, R. L., and Tarjan, R. E. Time bounds for selection. *Journal of Computer and System Sciences*, 7:448–461, 1973.
- Burri, N., von Rickenbach, P., and Wattenhofer, R. Dozer. Ultra-low power data gathering in sensor networks. In *International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.
- Chin, F. Y. L. and Ting, H. F. An improved algorithm for finding the median distributively. *Algorithmica*, 2:77–86, 1987.

- Frederickson, G. N. Tradeoffs for selection in distributed networks. In *Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 154–160, 1983.
- Goel, A. and Estrin, D. Simultaneous optimization for concave costs: Single sink aggregation or single source buy-at-bulk. *Algorithmica*, 43(1–2):5–15, 2005.
- Jia, L., Lin, G., Noubir, G., Rajaraman, R., and Sundaram, R. Universal approximations for TSP, Steiner Tree, and set cover. In *37th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 386–395, 2005.
- Kempe, D., Dobra, A., and Gehrke, J., Gossip-based computation of aggregate information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 2003.
- Madden, S., Franklin, M. J., Hellerstein, J. M., and Hong, W. TAG: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th Annual Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 131–146, 2002.
- Marberg, J. M. and Gafni, E. An optimal shout-echo algorithm for selection in distributed sets. In *Proceedings of the 23rd Allerton Conference on Communication, Control, and Computing*, 1985.
- Moscibroda, T. The worst-case capacity of wireless sensor networks. In *6th International Conference on Information Processing in Sensor Networks (IPSN)*, 2007.
- Moscibroda, T. and Wattenhofer, R. The complexity of connectivity in wireless networks. In *25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2006.
- Negro, A., Santoro, N., and Urrutia, J. Efficient distributed selection with bounded messages. *IEEE Transactions of Parallel and Distributed Systems*, 8(4):397–401, 1997.
- Patt-Shamir, B. A note on efficient aggregate queries in sensor networks. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 283–289, 2004.
- Peleg, D. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics and Applications, 2000.
- Rodeh, M. Finding the median distributively. *Journal of Computer and System Science*, 24(2):162–166, 1982.
- Rodem, D., Santoro, N., and Sidney, J. Shout-echo selection in distributed files. *Networks*, 16:235–249, 1986.
- Santoro, N., Scheutzw, M., and Sidney, J. B. On the expected complexity of distributed selection. *Journal of Parallel and Distributed Computing*, 5(2):194–203, 1988.
- Santoro, N., Sidney, J. B., and Sidney, S. J. A distributed selection algorithm and its expected communication complexity. *Theoretical Computer Science*, 100(1):185–204, 1992.
- Schönhage, A., Paterson, M. S., and Pippenger, N. Finding the median. *Journal of Computer and System Sciences*, 13:184–199, 1976.
- Shrira, L., Francez, N., and Rodeh, M. Distributed k -selection: From a sequential to a distributed algorithm. In *Proceedings of the 2nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 143–153, 1983.
- Tiwari, P. Lower bounds on communication complexity in distributed computer networks. *Journal of the ACM (JACM)*, 34(4):921, 1987.
- Yao, A. Some complexity questions related to distributive computing. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pp. 209, 1979.
- Yao, Y. and Gehrke, J. The Cougar approach to in-network query processing in sensor networks. *ACM SIGMOD Record*, 31(3):9–18, 2002.
- Zhao, J., Govindan, R., and Estrin, D. Computing aggregates for monitoring wireless sensor networks. In *Proceedings of the 1st IEEE International Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

Fabian Kuhn (kuhn@inf.ethz.ch) Postdoc researcher, Institute of Theoretical Computer Science, ETH Zurich, Switzerland

Thomas Locher (lochert@tik.ee.ethz.ch) PhD student, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland

Roger Wattenhofer (wattenhofer@tik.ee.ethz.ch) Professor, Head of Distributed Computing Group, Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland

© 2008 ACM 0001-0782/08/0900 \$5.00

CAREERS

Brooklyn College

Department of Computer & Information Science
Multimedia Computing Professor

The Department of Computer and Information Science is seeking a specialist in multimedia computing to direct a new undergraduate program in Multimedia Computing. This tenure-track position is at the rank of Assistant or Associate Professor, for appointment beginning September 1, 2009. The successful candidate will teach introductory and advanced undergraduate and/or graduate courses in graphics, gaming, robotics, multimedia and Internet technologies, and/or multimedia production, as well as other courses in computer science; will conduct research in the field; and will seek grant funding. The individual selected will also help develop multimedia courses and curricula for the Department and for the interdisciplinary Program in Performance and Interactive Media Arts. The successful candidate will have a Ph.D. and will have expertise in multimedia computing. For an Associate Professorship, the candidate should have a record of grant procurement. Candidates should also have a broad acquaintance with the field of computer science and should have good teaching skills. We are seeking candidates who are committed to undergraduate and graduate education at a public, urban institution that serves a highly diverse student body. Please send a cover letter, a c.v., statements of teaching and research philosophy, and evidence of good teaching. Please also have 3 reference letters sent. All materials should be sent to: Professor Aaron Tenenbaum, Chair, Department of Computer and Information Science, Brooklyn College, 2900 Bedford Avenue, Brooklyn, NY 11210, or via email to tbaum@sci.brooklyn.cuny.edu.

Elizabethtown College

Assistant or Associate Professor of Computer Science

Elizabethtown College invites applications for a tenure-track position in Computer Science, beginning August 2009. A Ph.D. in Computer Science or related field at the time of appointment is required. Duties include teaching of undergraduate Computing courses at both introductory and advanced levels, academic advising, scholarship, and institutional service (e.g., on faculty governance committees). Appointment at the Associate level is possible for a very well-qualified, experienced individual. Computing Science at Elizabethtown affirms the College's commitment to breadth in academic offerings, with well-established Bachelor of Science programs in Computer Science, Information Systems, and Computer Engineering (which, along with other Engineering programs, is currently a candidate for ABET accreditation), and minors in Computer Science and Information Systems. The relatively small size of the College en-

courages close cooperation among disciplines; in particular, there is a history of Computing working cooperatively with many other disciplines, including (but not limited to) its traditional cognates of Mathematics, Business, and Engineering. The successful candidate will have a strong commitment to undergraduate teaching and advising in an academically stimulating, supportive environment that offers a wide range of curricular options. Specifically, although some research specialization is expected in candidates, teaching excellence in multiple areas of Computing, and demonstrated understanding of Computing as a discipline of considerable breadth, will weigh more heavily than narrowly-focused research strength in one particular sub-discipline. Furthermore, given the close, cooperative relationship between Computing and Engineering, and the ABET accreditation candidacy of the latter, it is hoped that individuals having documented familiarity with Engineering will find this opportunity particularly interesting. Located in southeastern Pennsylvania, Elizabethtown College offers its 2,000 students more than 47 academic programs in the liberal arts, sciences and professional studies. Driven by its motto to "Educate for Service," Elizabethtown centers learning in strong relationships, links classroom instruction with experiential learning, emphasizes international and cross-cultural perspectives and nurtures the capacity for lives of purpose and leadership as global citizens. For more information, consult <http://www.etown.edu/>. To apply, candidates should send a letter of interest, contact information, a CV that includes a list of courses taught, and the names and contact information of at least three references to: Elizabethtown College, Attn: Human Resources, One Alpha Drive, Elizabethtown, PA 17022-2298hr@etown.edu For more information visit: www.etown.edu/humanresources Review of applications will begin October 1, 2008. AA/EOE

Japan Advanced Institute of Science and Technology Associate Professor

JAIST invites applicants to an associate professor position in SCHOOL OF INFORMATION SCIENCE. The appointee is expected to start his academic and educational activities in JAIST no later than April of 2009.

Candidates have to be highly competent in conducting research work in the areas of Ubiquitous Computing and Networking technologies. Deep knowledge bases in Logic Circuit and Computer Architecture, Communication System and Communication Protocols, and their related technologies are expected.

OBJECTIVE: A primary objective of this position is to promote domestic and international research and development activities in reliable information and communication technologies.

QUALIFICATION: Applicants have to hold Ph.D degree, and be qualified for high scientific activities through participating in domestic and international research initiatives. The search committee may ask candidates to demonstrate teaching and pedantic skills by providing demonstration lectures in Japanese as well as in English.

SELECTION: The search committee shall evaluate the candidates' expertise, research activities, and teaching skills. The procedure for the evaluation is strictly impartial, unbiased, and fair, but the evaluation result of the each applicant will not be released after the selection. Names and contact information of references should be included in the application documents. The search committee may contact the references and ask for detailed information about the candidates. If the qualifications of the candidates are equivalent, the search committee shall prioritize women and/or foreigners.

Applicants must submit some documents: Details can be found at: http://www.jaist.ac.jp/jimu/syomu/koubo/network_AP-e.htm

Deadline: Applications must be received no later than Dec. 1, 2008

More detailed information can be found at: <http://www.jaist.ac.jp/is/index.html>
National University of Singapore

Kwan Im Thong Hood Cho Temple Professorship

The School of Computing (SoC) at the National University of Singapore (NUS) invites applications and nominations for the Kwan Im Thong Hood Cho Temple (KITHCT) Professorship in Computing. (This Chair is endowed by a temple for the Bodhisattva of Compassion or Goddess of Mercy.)

We seek a renowned scholar at the full Professor level. The successful applicant is expected to lead an internationally-recognized research group in SoC. The salary and benefits are internationally competitive. The preferred start date is January, 2009.

NUS (<http://www.nus.edu.sg/>) is a research university, with low teaching loads, excellent facilities and ample research funding (both within NUS and through funding agencies).

SoC (<http://www.comp.nus.edu.sg/>) has about 100 faculty members in two departments: Computer Science (CS) and Information Systems (IS). They regularly publish in prestigious conferences and journals, as well as serve on their program committees and editorial boards.

To apply, please send a resume, statements on research, teaching and leadership, and six references to: Prof. Y.C. Tay, KITHCT Search Committee Chair, School of Computing, National University of Singapore, Singapore 117590, Republic of Singapore (Attn: SoC HR Office, E-mail: hr-office@comp.nus.edu.sg).

University of New Brunswick Faculty of Computer Science

The Faculty of Computer Science, University of New Brunswick invites applications for a tenure-track position at the rank of Assistant Professor. Applicants must have a PhD in Computer Science by the starting date of the appointment.

We invite applications from candidates whose focus is in Networking and Data Communication or Human Computer Interaction. The successful candidate must have a strong commitment to excellence in research and teaching, be able to teach courses at the undergraduate and graduate level and supervise graduate students.

Subject to budgetary approval, this position will be effective as soon as January 1, 2009. Salary levels will be commensurate with qualifications and experience.

Applicants should send curriculum vitae, the names and addresses of three referees, a one-page statement of research interests, a one-page teaching philosophy, and copies of three relevant technical publications to:

Dr. Ali A. Ghorbani, Dean
Faculty of Computer Science
University of New Brunswick
Fredericton, New Brunswick Canada
Fax # 506-453-3566 email: ghorbani@unb.ca

The search committee will begin to review applications on October 15, 2008 and will continue to do so until the position is filled. For more information on the Faculty of Computer Science, visit <http://www.cs.unb.ca/>.

The University of New Brunswick is committed to employment equity and encourages applications from qualified women and men, visible minorities, aboriginal people and persons with disabilities. All qualified applicants are encouraged to apply; however Canadian citizens and permanent residents will be given priority.

University of Pennsylvania Department of Computer and Information Science Faculty Positions

The University of Pennsylvania invites applicants for tenure-track appointments in both experimental and theoretical computer science to start July 1, 2009. Tenured appointments will also be considered. Faculty duties include teaching undergraduate and graduate students and conducting high-quality research.

The Department of Computer and Information Science has undergone a major expansion, including new faculty positions and a new building, Levine Hall, which was opened in April 2003. Over the last few years, we have successfully recruited faculty in artificial intelligence, architecture, databases, machine vision, programming languages, security and graphics. We are now especially interested in candidates in architecture and systems, although outstanding candidates in other areas might also be considered. Successful applicants will find Penn to be a stimulating environment conducive to professional growth.

The University of Pennsylvania is an Ivy League University located near the center of Philadelphia, the 5th largest city in the US. Within walking distance of each other are its Schools of Arts and

Sciences, Engineering, Medicine, the Wharton School, the Annenberg School of Communication, Nursing, Law, and Fine Arts. The University campus and the Philadelphia area support a rich diversity of scientific, educational, and cultural opportunities, major technology-driven industries such as pharmaceuticals, finance, and aerospace, as well as attractive urban and suburban residential neighborhoods. Princeton and New York City are within commuting distance.

To apply, please complete the form located on the Faculty Recruitment Web Site at:

<http://www.cis.upenn.edu/departamental/facultyRecruiting.shtml>

Electronic applications are strongly preferred, but hard-copy applications (including the names of at least four references) may alternatively be sent to:

Chair, Faculty Search Committee
Department of Computer and Information
Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

Applications should be received by January 15, 2009 to be assured full consideration.

Applications will be accepted until positions are filled.

Questions can be addressed to faculty-search@central.cis.upenn.edu.

The University of Pennsylvania values diversity and seeks talented students, faculty and staff from diverse backgrounds. The University of Pennsylvania does not discriminate on the basis of race, sex, sexual orientation, gender identity, religion, color, national or ethnic origin, age, disability, or status

as a Vietnam Era Veteran or disabled veteran in the administration of educational policies, programs or activities; admissions policies; scholarship and loan awards; athletic, or other University administered programs or employment.

The Penn CIS Faculty is sensitive to "two-body problems" and would be pleased to assist with opportunities in the Philadelphia region.

Universidad Politecnica de Madrid Associate Prof. Tenure track (Prof. Titular Universidad interino)

The Department of Artificial Intelligence of the Universidad Politecnica de Madrid invites applicants for a tenure track position at the level of Associate Professor ("Profesor Titular de Universidad interino") starting in 2009. Applicants with the initiative to start new and ground-breaking research activities which complement existing work in the department are sought. Preference will be given to candidates whose research relates to areas covered by the department, but candidates in all areas of computer science are encouraged to apply. An initial two year contract will be offered, with the possibility for later permanent appointment at the associate professor level ("Profesor Titular de Universidad"). Responsibilities will include teaching undergraduate and graduate courses related to the applicant's area of expertise and conducting high-level research.

Candidates must have a PhD in Computer Science (or a closely related discipline), be citizens of the European Union (as it is a Spanish Civil Servant appointment), and have fluent English or Spanish skills.



Windows Kernel Source and Curriculum Materials for Academic Teaching and Research.

The Windows® Academic Program from Microsoft® provides the materials you need to integrate Windows kernel technology into the teaching and research of operating systems.

The program includes:

- **Windows Research Kernel (WRK):** Sources to build and experiment with a fully-functional version of the Windows kernel for x86 and x64 platforms, as well as the original design documents for Windows NT.
- **Curriculum Resource Kit (CRK):** PowerPoint® slides presenting the details of the design and implementation of the Windows kernel, following the ACM/IEEE-CS OS Body of Knowledge, and including labs, exercises, quiz questions, and links to the relevant sources.
- **ProjectOZ:** An OS project environment based on the SPACE kernel-less OS project at UC Santa Barbara, allowing students to develop OS kernel projects in user-mode.

These materials are available at no cost, but only for non-commercial use by universities.

For more information, visit www.microsoft.com/WindowsAcademic
or e-mail compsci@microsoft.com.

Faculty and Academic Staff Positions
College of Engineering
Alfaisal University



Alfaisal University is a private, non-profit, research university comprising of the Colleges of Engineering, Science, Medicine and Business. The language of instruction is English and modern learning outcomes, paradigms and technologies are used. The university was founded by The King Faisal Foundation along with organizations such as Boeing, BAE Systems, United Technologies, THALES and King Faisal Specialist Hospital & Research Center, who all serve on its Board of Trustees.

The College of Engineering will offer undergraduate and graduate programs in the following disciplines and areas: **ELECTRICAL** (power, communications, signal processing, electronics, photonics, remote sensing and geodata analysis), **COMPUTER** (intelligent systems, language and speech, computer systems, computation), **MECHANICAL** (applied mechanics, thermo/fluid engineering, product creation), **AEROSPACE** (propulsion, aerospace systems, transportation, system dynamics and control), **MATERIALS** (materials processing, materials properties and performance, polymers, nanoscience and technology), **CHEMICAL** (catalysis, reactor design, separations, design-systems), **INDUSTRIAL** (operations research, product and operations management, engineering management economics and finance). All programs have been developed by renowned scholars from leading universities in the US and the UK and are designed to be qualified for accreditation according to US and UK educational standards.

Alfaisal Engineering seeks candidates for the following positions: **SENIOR FACULTY** (with research, instructional, and administrative responsibilities), **JUNIOR FACULTY** (with research and instructional responsibilities). Attractive salary and start-up support is provided. Queries and applications should be sent to dean_engnr@alfaisal.edu. The subject line should specify the discipline, area, position and the announcement reference. The deadline for applications is 15th October, 2008. Interviews for leading candidates will be conducted from 15th to 20th December, 2008 in Cambridge, MA, USA.

RESEARCH SCIENTISTS (academics with research focus), **LECTURERS** (academics with instructional focus), **POST-DOCS** (Doctoral degree holders with research focus), **INSTRUCTORS** (Master's degree holders with instructional focus) and **ENGINEERS** (Bachelor's degree holders). Queries and applications should be sent to engnr_academic@alfaisal.edu. The subject line should specify the discipline, area, position and the announcement reference. The deadline for applications is 15th October, 2008.

Alfaisal University | P.O.Box 50927 | Riyadh 11533 | Saudi Arabia

www.alfaisal.edu



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to jonathan.just@acm.org. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.

Estimates: An insertion order will then be e-mailed back to you. The ad will be typeset according to CACM guidelines. **NO PROOFS** can be sent. Classified line ads are **NOT** commissionable.

Rates: \$295.00 for six lines of text, 40 characters per line. \$80.00 for each additional three lines. The **MINIMUM** is six lines.

Deadlines: Five weeks prior to the publication date of the issue (which is the first of every month). Latest deadlines: <http://www.acm.org/publications>

Career Opportunities Online: Classified and recruitment display ads receive a free duplicate listing on our website at: <http://campus.acm.org/careercenter>

Ads are listed for a period of six weeks.

For More Information Contact:

JONATHAN JUST
Director of Media Sales
at 212-626-0687 or
jonathan.just@acm.org

Assistant Professors, Department of Computer Science



UNIVERSITY OF
CALGARY

The Department of Computer Science, Faculty of Science, at the University of Calgary seeks outstanding candidates for several tenure-track positions at the Assistant Professor level. Applicants from Information Security, Theory, HCI/Information Visualization, and Computer Games are of particular interest. Details for each position appear at: <http://www.cpsc.ucalgary.ca>. Applicants must possess a doctorate in Computer Science or a related discipline at the time of appointment, and have a strong potential to develop an excellent research record.

The Department is one of Canada's leaders as evidenced by our commitment to excellence in research and teaching. It has an expansive graduate program and extensive state-of-the-art computing facilities. Calgary is a multicultural city that is the fastest growing city in Canada. Calgary enjoys a moderate climate located beside the natural beauty of the Rocky Mountains. Further information about the Department is available at <http://www.cpsc.ucalgary.ca>.

Interested applicants should send a CV, a concise description of their research area and program, a statement of teaching philosophy, and arrange to have at least three reference letters sent to: Dr. Frank Maurer, Department of Computer Science, University of Calgary, Calgary, Alberta, Canada, T2N 1N4 or via email to search@cpsc.ucalgary.ca. The applications will be reviewed beginning November 2008 and will continue until the positions are filled.

All qualified candidates are encouraged to apply; however, Canadians and permanent residents will be given priority.



DOI:10.1145/1378727.1389570

Peter Winkler

Puzzled Solutions and Sources

Last month (August 2008, p. 104) Peter Winkler posed a trio of brain teasers in his “Puzzled” column. Here, he offers some solutions. How well did you do?

1. Election Year

Solution: To prove that the opinions eventually become fixed or cycle every other day, think of each acquaintance-ship between citizens as a pair of arrows, one in each direction. Let us say an arrow is currently “bad” if the party of the citizen at its tail differs from the next-day’s party of the citizen at its head.

Consider the arrows pointing out from citizen Clyde on day t , during which (say) Clyde is a Democrat. Suppose that m of these arrows are bad. If Clyde is still (or is again) a Democrat on day $t+1$, then the number n of bad arrows pointing toward Clyde at day t will be exactly m .

If, however, Clyde is a Republican on day $t+1$, n will be strictly less than m since it must have been that the majority of his friends were Republican on day t . Therefore a majority of the arrows out of Clyde were bad on day $t-1$, and now only a minority of the arrows into Clyde on day t are bad.

The same observations hold if Clyde is a Republican on day $t-1$.

But, here’s the key: Every arrow is out of someone on day $t-1$ and into someone on day t . Thus, the total number of bad arrows cannot increase between days $t-1$ and t ; in fact, that number will go down unless every citizen had the same opinion on day $t-1$ as on day $t+1$.

But the total number of bad arrows on a given day cannot decrease forever and must eventually reach some number k from which it never descends. At that point, every citizen will either

stick with his or her party forever or flop back and forth every day.

The puzzle can be generalized considerably by, for example, adding weights to vertices (meaning some citizens’ party memberships are more prized than others), allowing loops (citizens who consider their own current opinions as well), and allowing tie-breaking mechanisms and even different thresholds for party changes.

Source: This puzzle was suggested to me by Sasha Razborov of the Institute for Advanced Study in Princeton, NJ, who says it was considered for an International Mathematics Olympiad but rejected as too difficult. It was posed and solved in a paper by Goles, E. and Olivos, J. Periodic behavior of generalized threshold functions. *Discrete Mathematics* 30 (1980), 187–189.

2. Island of Foosgangland

Solution: Suppose there are n intersections and m paths in Foosgangland; we may assume the paths are numbered 1 through m . Since each path has two ends, the average number of paths meeting at an intersection is $2m/n$.

Let us place a pedestrian at every intersection. At time 1, the pedestrians at each end of path 1 change places, giving a friendly wave as they pass each other in the middle of the path. At time 2, the pedestrians currently at each end of path 2 change places. This continues until at time m , the pedestrians who find themselves at each end of path m change places.

What has happened here? Observe first that every one of the n pedestrians has taken an “increasing walk”; that is, he or she has walked a path, rested, then walked a higher-numbered path,

and so forth. Since every edge has been traversed twice, the total length of all these walks is $2m$. But then the average length of the walks is $2m/n$, and it follows that at least one of the paths has length at least $2m/n$, and we are done.

Source: This puzzle and its elegant solution were passed to me by Ehud Friedgut of The Hebrew University of Jerusalem. We traced the puzzle back to the fourth problem in the second round of the 1994 Bundeswettbewerb Mathematik, one of two major German mathematical competitions.

3. Graph Coloring Game

This puzzle remains unsolved as of this writing. We have no idea how to go about it. The usual approaches (such as symmetrization, strategy stealing, and induction) don’t seem to work. But surely it should be true, don’t you think?

Peter Winkler (puzzled@caom.acm.org) is Professor of Mathematics and of Computer Science and Albert Bradley Third Century Professor in the Sciences at Dartmouth College, Hanover, NH. He has written two puzzle books: *Mathematical Puzzles: A Connoisseur’s Collection* and *Mathematical Mind-benders*, both published by A K Peters, Ltd., Wellesley, MA.

Future Tense, one of the revolving features on this page, presents stories from the intersection of computational science and technological speculation, their boundaries limited only by our ability to imagine what will and could be.

DOI:10.1145/1378727.1378750

William Sims Bainbridge

Future Tense Will

Expect virtual immortality through enduring, realistic avatars based on published work and archived memory.

WHEN I SWITCHED ON the computer simulation of my long-dead grandfather, Dr. William Seaman Bainbridge (1870–1947), I rather expected he would be irrational, incoherent, or even insane. Thus his avatar surprised me when he looked me coolly in the eye and ordered me, in a commanding tone of voice, to explain his situation. I should have realized that a man capable of dissecting his own father's skull, hoping to explain his parents' estrangement as the result of a brain lesion, would not be squeamish. I should have remembered that his nickname was Will.

I told him I had constructed an NLP/HMM/CBR/AI avatar with information about his life and thoughts, including his 11 semiautobiographical medical books and 100 journal articles in which he described the surgery he had performed, his thoughts about psychiatry, and his principles for good health. Other aspects of his personality had come from personal letters, travel diaries (1918–1941), and the childhood experiences his parents described in the books they wrote about their 1879–1880 world tour of Protestant missions. For the kinesics of his avatar, I extracted motion-capture sequences from home movies. For the dialogue system, the best I could do was take an average of two men whose recordings matched my dim memory of his voice: U.S. President Teddy Roosevelt and Will's cousin, Bainbridge Colby, who had been President Woodrow Wilson's secretary of state. Gently, I explained that while his wife and children had all died, many grandchildren, great-

grandchildren, and even great-great-grandchildren still lived.


His avatar stood motionless for a moment, then looked down, moved his hand across his simulated face as if to brush away real tears, and sighed. "I remember nothing of the afterlife," he said. "Does that mean my religious faith was untrue?" I assured him it was perfectly natural that he could not recall Heaven, because all his data came from before his death. We even knew his last faith-filled words to his wife, because his friend, Reverend Norman Vincent Peale, had published them in a book: "June, I shall wait for you on the other side." I suggested to him that their souls had been united now for many decades. I did not tell him that June had died only after Alzheimer's had apparently robbed her of all memory of him.

After deep thought, he straightened up, winked at me, and cheerfully exclaimed, "Then we have much work to do!" Scientist that he was, he bombarded me with increasingly more sophisticated technical questions until he had a full grasp of the process I had used to emulate him. He demanded that I show him modern information technologies, so I gave him a tour of the tablet computer on which I was taking notes. When I somewhat condescendingly explained email to him, he halted my monologue with a harsh stare, saying, "But I used email in 1923 to communicate with my mother while I was investigating the Ruhr crisis in Germany!" I began to protest that this was impossible, but he wagged his virtual finger and told me that his cable address had been "bridgebain new york," and that

a telegram sent from anywhere in the world would arrive at his office instantly, in the hands of the Western Union boy. Further, he proclaimed angrily, he had proposed marriage to June back in 1911 via long-distance telephone, so he was not the ignorant savage I apparently supposed he was.

Before I could think of excuses, he compelled me to begin simulating his mother, Lucy. Although she had died in 1928 and left neither movies nor memories in the mind of any living person, she had published several books describing her world travels and her 16 years helping immigrants on behalf of the New York City Mission Society. This organization was still active, and its storeroom held her scrapbooks plus copies of her monthly essays for the Mission Society magazine. Will said he had selected his mother for the second emulation experiment simply because her thoughts were well documented, but I knew better. For years after her death, he had aggressively promoted her fame, forcing one of his patients to write a sanctimonious biography, and presented copies of her books to everybody from Helen Keller to the queen of Belgium.

The full scope of my fate then dawned on me. Once Lucy had been resurrected, they would force me to restore her husband who also had published extensively. He had separated from her in 1890, so they would adjust his avatar to make him more docile. Will's favorite cousin, "Bain" Colby, would be next, then all the other deceased writers in the family.

I need to find Will a high-paying job before he drives me into bankruptcy, but now I am fully committed to realizing his dream. He promises to resurrect me, after I myself expire. 

William Sims Bainbridge (wsbainbridge@gmail.com) is a sociologist, computer programmer, and information scientist who does research on technological innovation, religious movements, and virtual worlds; he currently directs Human-Centered Computing in the computer science directorate of the National Science Foundation, Arlington, VA.



Symposium on
**Computer Human Interaction for
Management of Information Technology**

November 14/15, 2008 San Diego, CA

A Turning Point in IT

General Chairs:

*A Eelen Frisch, Exponential
Eser Kandogan, IBM Research*

Program Chairs:

*Wayne Lutters, UMBC
Jim Thornton, PARC
Mustapha Mouloua, UCF*

Steering Committee:

*Stephen Barley, Stanford
David Blank-Edelman,
Northeastern
Jack Carroll, Penn State
Alva Couch, Tufts
Patricia Jones, NASA Ames
Rob Kolstad
Paul Maglio, IBM Research
Tom Sheridan, MIT*

Publicity

*George Engelbeck, Microsoft
Nate Gunderson, Microsoft*

Dates

Papers Submission

April 7th, 2008

**Posters and
Experience**

Reports

May 30th, 2008

Invitation to Participate

CHIMIT is the leading forum for discussing topics on IT management with a focus on people, business, and technology. Researchers and practitioners are invited to share issues, solutions, and research drawing upon fields such as human-computer interaction, human factors, computer systems, and management and service sciences.

Topics

Original unpublished contributions are sought in areas including but not limited to:

Workspace Studies

- Ethnographic studies of IT work in context
- Patterns of work for various IT processes

Processes and Practices

- Development and use of processes in IT
- Impact of business decisions on IT

Organizational Knowledge

- Studies of collaboration and coordination
- Knowledge management and training

Design

- Design of human-centered IT systems
- Architectural considerations for user experience

Tools and Techniques

- Collaborative system administration workspaces
- Visualizations of complex system behavior

Automation

- Automation/Policy languages
- Human interfaces to automation

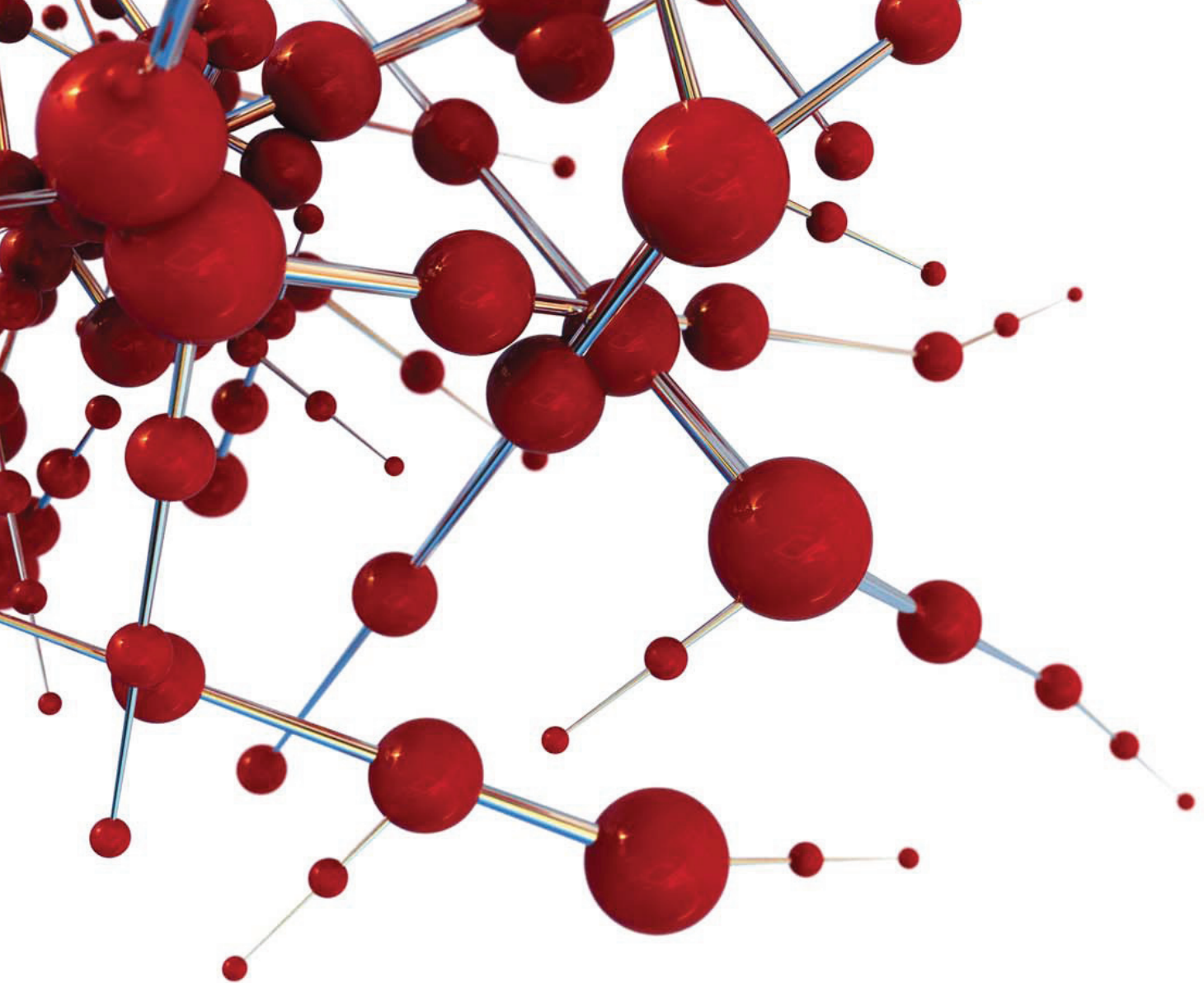


In cooperation with

USENIX

www.chimit08.org

November 14/15, 2008 | San Diego, CA



**CONNECT WITH OUR
COMMUNITY OF EXPERTS.**

www.reviews.com



Association for
Computing Machinery

Reviews.com

They'll help you find the best new books
and articles in computing.

Computing Reviews is a collaboration between the ACM and Reviews.com.