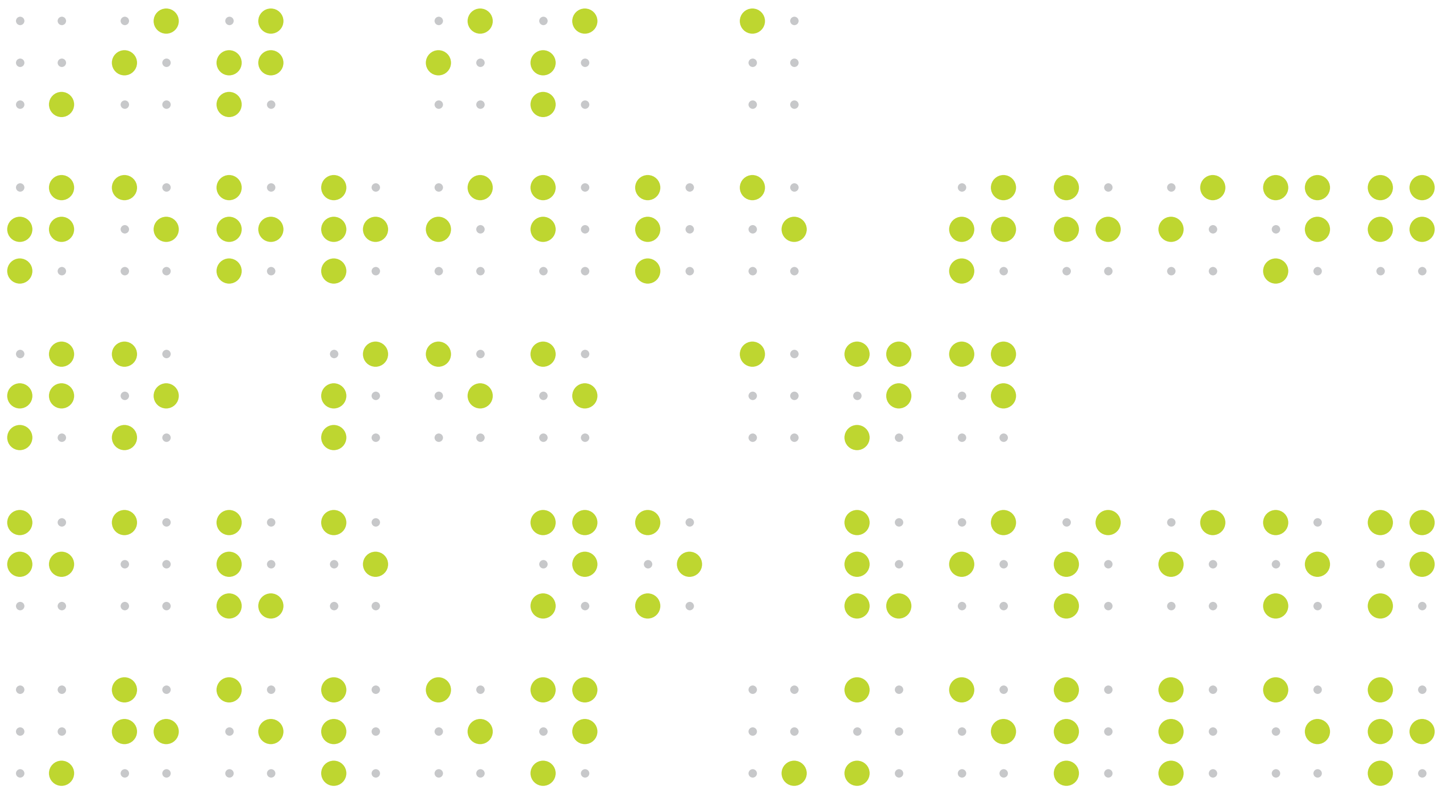
COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

08/09 VOL.52 NO.08

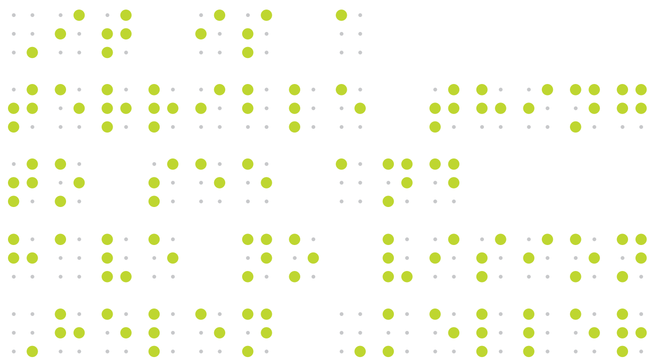


A Blind Person's Interaction with Technology

The Pathologies of Big Data

Security Lessons from Google Chrome

Recommender Systems



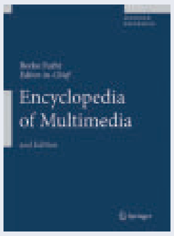
It is a terrible thing
to see and have no vision.
Helen Keller

Springer
the language of science

springer.com

SPRINGER REFERENCE

Springer References on Computer Science



Encyclopedia of Multimedia

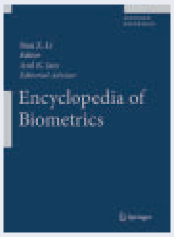
B. Furht (Ed.)

Encyclopedia of Multimedia, Second Edition provides easy access to important concepts, issues and technology trends in the field of multimedia technologies, systems, techniques, and applications. Hundreds of leading researchers and world experts have contributed to this comprehensive collection of nearly 400 entries. Over 1200 heavily-illustrated pages (including 120+ new entries) present concise overviews.

Hardcover
2nd ed. 2008. XXX, 1001 p. 565 illus.
ISBN 978-0-387-74724-8 ► **$449.00**

eReference
2nd ed. 2008.
ISBN 978-0-387-78414-4 ► **$449.00**

Print + eReference
2nd ed. 2008. XXX, 983 p. 565 illus.
ISBN 978-0-387-78415-1 ► **$559.00**



Encyclopedia of Biometrics

S. Z. Li (Ed.)

Advisory editor: **A. K. Jain**

Biometrics refers to automated methods of recognizing a person based on physiological or behavioral characteristics. The Encyclopedia of Biometrics provides a comprehensive reference to topics in Biometrics, including concepts, modalities, algorithms, devices, systems, security, performance testing, applications and standardization.

Hardcover
2009. XXXII, 1432 p. 680 illus. (In 2 volumes, not available separately)
ISBN 978-0-387-73002-8 ► **$449.00**

eReference
2009.
ISBN 978-0-387-73003-5 ► **$449.00**

Print + eReference
2009. XXXII, 1432 p. 680 illus. (In 2 volumes, not available separately)
ISBN 978-0-387-73004-2 ► **$559.00**

Encyclopedia of Parallel Computing

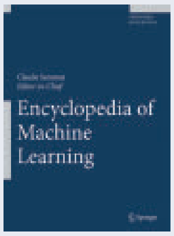
Editor-in-chief: **D. Padua**

The Encyclopedia of Parallel Computing is broad in scope, covering machine organization, programming, algorithms, and applications. Within each area, the Encyclopedia covers concepts, designs, and specific implementations. The area of algorithms covers concepts such as cache-oblivious algorithms and systolic algorithms; specific numerical and non-numerical algorithms such as parallel matrix-matrix multiplication and graph algorithms; and implementations of algorithms in the form of widely used libraries such as LAPACK.

Hardcover
2011. Approx. 2010 p. (In 3 volumes, not available separately)
ISBN 978-0-387-09765-7 ► **approx. $999.00**

eReference
2011.
ISBN 978-0-387-09766-4 ► **approx. $999.00**

Print + eReference
2011. Approx. 2010 p. (In 3 volumes, not available separately)
ISBN 978-0-387-09844-9 ► **approx. $1250.00**



Encyclopedia of Machine Learning

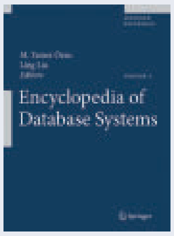
Editor-in-chief:
C. Sammut, G. I. Webb

This comprehensive encyclopedia, in A-Z format, provides easy access to relevant information for those seeking entry into any aspect within the broad field of Machine Learning. Most of the several hundred entries in this preeminent work include useful literature references, providing the reader with a portal to more detailed information on any given topic.

Hardcover
2010. Approx. 1000 p.
ISBN 978-0-387-30768-8 ► **approx. $499.00**

eReference
2010.
ISBN 978-0-387-30164-8 ► **approx. $499.00**

Print + eReference
2010. Approx. 1000 p.
ISBN 978-0-387-34558-1 ► **approx. $629.00**



Encyclopedia of Database Systems

M. T. Özsu (Eds.)

The Encyclopedia of Databases, a comprehensive work, provides easy access to relevant information on all aspects of very large databases. This volume features alphabetical organization of concepts covering main areas of very large databases. These 1000 entries offer convenient access to information in the field of databases with definitions and illustrations of basic terminology, concepts, methods, and algorithms, references to literature, and cross-references to other entries and journal articles. Topics for the encyclopedia were selected by a distinguished international advisory board, and written by world class experts in the field. The Encyclopedia of Databases is designed to meet the needs of research scientists, professors and graduate-level students in computer science and engineering. This encyclopedia is also suitable for practitioners in industry.

Hardcover
2009. Approx. 4000 p. 60 illus. (In 5 volumes, not available separately)
ISBN 978-0-387-35544-3 ► **$1499.00**

eReference
2009.
ISBN 978-0-387-39940-9 ► **$1499.00**

Print + eReference.
2009. Approx. 4000 p. 60 illus. (In 5 volumes, not available separately)
ISBN 978-0-387-49616-0 ► **$1899.00**

Easy Ways to Order for the Americas ► **Write:** Springer Order Department, PO Box 2485, Secaucus, NJ 07096-2485, USA ► **Call: (toll free)** 1-800-SPRINGER ► **Fax:** 1-201-348-4505 ► **Email:** orders-ny@springer.com or **for outside the Americas** ► **Write:** Springer Customer Service Center GmbH, Haberstrasse 7, 69126 Heidelberg, Germany ► **Call:** +49 (0) 6221-345-4301 ► **Fax :** +49 (0) 6221-345-4229 ► **Email:** orders-hd-individuals@springer.com
► Prices are subject to change without notice. All prices are net prices.

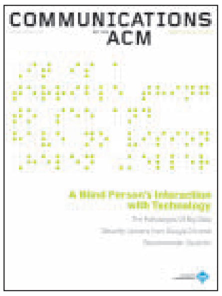
014089x

COMMUNICATIONS OF THE ACM

ILLUSTRATION BY YAREK WASZUL

Association for Computing Machinery
Advancing Computing as a Science & Profession

08/2009
VOL. 52 NO. 08



About the Cover: One of the most effective methods for designing tools and technologies for blind users is to observe how they interact and go about their daily lives, say Kristen Shinohara and Josh Tenenberg in their article on page 58. The cover represents a famous quote from Helen Keller, as explained on the inside front cover. The symbols, though much larger than readable Braille, were translated into Braille by Alicia Kubista using an online Braille generator.

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.

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.

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
Jennifer Ruzicka
jen.ruzicka@hq.acm.org

Media Kit acmmediasales@acm.org

BPA Audit Pending

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://www.acm-media.org 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.

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/1536616.1536617

Steve Bourne and Bryan Cantrill

Communications and the Practitioner

A year ago, this publication was stripped to the studs and rebuilt, with everything from the content to the cover art revisited, rethought, and revitalized. Among the many additions has been a new section, "Practice." While readers may have not recognized this section per se, they have likely noticed its practitioner-oriented content, with articles on everything from the innards of GPUs and the mechanics of hard-drive failure to debugging AJAX and avoiding the contagious virulence of XML fever—and much in between.

The story behind the Practice section merits some explanation, for it traces the history of the practitioner within ACM. There was a time in the not-too-distant past when practitioners felt largely indifferent about ACM: while practitioners grappled with thorny problems posing non-negotiable obstacles to shipping a product or deploying a system, ACM (and, dare we say, its flagship publication) could seem to be comfortably insulated in dreamy abstraction. Not that the practitioner was better served by anyone else: much content for the professional software engineer seemed to be either explicitly "for dummies" or shamelessly capitalizing on the latest fad—and often both.

Several at ACM saw these two problems—ACM's lack of focus on the practitioner and the opportunity posed by the paucity of high-caliber practitioner content—and set out to address them with a new magazine. The result of this effort, *ACM Queue*, launched in March 2003 and was targeted toward the practitioner, but with an eye toward tomorrow's problems instead of today's solutions. In its first issue, *Queue* introduced itself as "a tonic for the hype weary, with a commitment to methodically dissect upcoming challenges while posing the same hard questions software developers ask themselves." Over the years, *Queue* stayed true to this vision by having editorial content conceived of and written by software engineers themselves. This afforded the magazine a problem focus, but one that held fast to the reality of production systems. Striking this delicate balance required constant vigilance, but it made *Queue* a must-read for leading practitioners.

With everything about *Communications* being reconsidered, the time was perfect to rethink not just the relationship between *Queue* and *Communications*, but also the role of the practitioner within ACM itself.

The growing success of *Queue* coincided with another change at ACM: the remaking of *Communications*. There had long been a desire among ACM's leadership to refresh the venerable publication, and with everything about *Communications* being reconsidered, the time was perfect to rethink not just the relationship between *Queue* and *Communications*, but also the role of the practitioner within ACM itself. From this deliberation, the new Practice section was born: *Queue* retained its identity and its Web site, but also became an integral part of the new *Communications*, with *Queue* leading the development of articles in the Practice section as well as of columns such as Kode Vicious. Bringing *Queue* content to *Communications* has reunited the practitioner and the researcher under a single masthead, reinvigorating both communities. Indeed, laying the best work of the practitioner and researcher communities side-by-side has been a refreshing reminder of what brings us together: we of ACM are united by our common passion for making computers do nifty and useful things.

So to longtime ACM members and *Communications* readers, we hope you have found the new Practice section to be thought-provoking. And to practitioners (especially, those who may be new ACM members!), we hope you will not restrict yourselves to the Practice section, but will also take the time to read the latest work from the larger ACM community. To everyone, welcome to the new *Communications* and the new broader ACM!

Steve Bourne is Chief Technology Officer, Eldorado Ventures, editor-in-chief of *ACM Queue*, chair of ACM's Professions Board, and chair of *Communications'* Practice Board.

Bryan Cantrill is a Distinguished Engineer at Sun Microsystems, member of *ACM Queue* Editorial Board, and member of ACM's Professions Board.

Take Advantage of *ACM's Lifetime Membership Plan!*

- ACM Professional Members can enjoy the convenience of making a single payment for their entire tenure as an ACM Member, and also be protected from future price increases by taking advantage of ACM's Lifetime Membership option.
- ACM Lifetime Membership dues may be tax deductible under certain circumstances, so becoming a Lifetime Member can have additional advantages if you act before the end of 2009. (Please consult with your tax advisor.)
- Lifetime Members receive a certificate of recognition suitable for framing, and enjoy all of the benefits of ACM Professional Membership.

Learn more and apply at:
http://www.acm.org/life

Association for Computing Machinery

Advancing Computing as a Science & Profession

DIARISE NOW!

ICSE CAPE TOWN 2010

32nd International Conference on Software Engineering
2 – 8 MAY 2010 CAPE TOWN, SOUTH AFRICA
Cape Town International Convention Centre

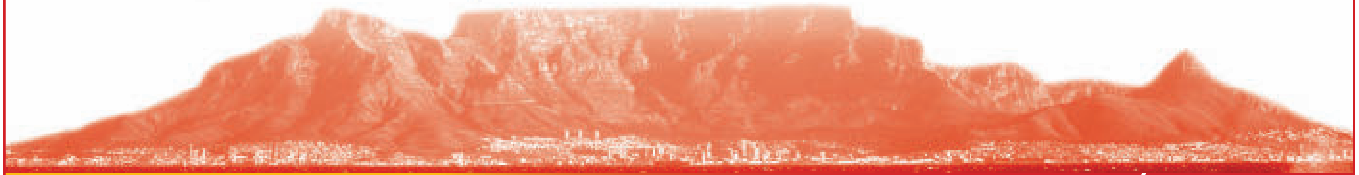
ICSE 2010 takes place in Cape Town, one of the world's most spectacular destinations, blessed with some of the most magnificent scenery and natural beauty imaginable and offering six of South Africa's top ten tourist attractions within 40 minutes drive of the city.

SIGSOFT IEEE computer society

Tracks
Technical/Research, SE in Practice, Education
New Ideas and Emerging Results

Events
Workshops, Tutorials, Research Demo's
Doctoral and Emerging Faculty Symposia

Organisation
General Chairs:
Jeff Kramer and Judith Bishop
Programme Chairs:
Sebastian Uchitel and Prem Devanbu
Conference Organiser:
SBS Conferences



www.sbs.co.za/icse2010

DOI:10.1145/1536616.1536618

Scott E. Delman

Responding to the Blogosphere

In the July issue of *Communications*, Moshe Vardi addressed ACM's access model in his Editor's Letter entitled "Open, Closed, or Clopen Access?" (p. 5). The letter has since been picked up in the blogosphere by John Dupuis, the esteemed Science & Engineering Librarian from York University in Toronto, on his blog *Confessions of a Science Librarian* (http://scienceblogs.com/confessions). Dupuis raised some interesting questions that I believe deserve response, especially since as a scholarly publisher of nearly two decades the topics of open access (OA) and association publishing are particularly near and dear to my heart.

It is worth repeating a sentiment that Dupuis offered in his blog that ACM is "on the side of the angels," not just because I really like the sound of this, but because the sentiment underscores an important truth related to both OA and association publishing, the truth that association publishers are in effect all OA publishers. In the hotly debated realm of OA, sides have been drawn and too often the issues and players are portrayed as black and white, right or wrong, good and evil. These dichotomies are easy to package and sell to those who are buying, but all too often they are inaccurate.

So is the case with the now famous color-coded guide to OA publishers. This system places publishers in boxes, but fails to address the reality that association publishers (regardless of their color-coded status) serve as field-wide gatekeepers of information and knowledge without a profit motive to drive their decision making. Associations like ACM are their membership, and as a result, are simply an extension of the intentions and will of the scientific communities they serve. Much like the notion of an institutional repository at a more targeted level, associations provide a single point of entry for members to access the historical and ongoing record of scholarship for their entire field (if executed well, that is).

The fact that ACM charges both for access to the published information in its Digital Library and also extends the courtesy of "Green OA" to its authors is actually less important to me (while both are important aspects of what we do) than the fact that ACM and many other association publishers serve as well-intentioned caretakers of the scholarly record. I have spent too many hours trying to identify the "most up-to-date version" of an author's article on his or her Web site or digging through the various related institutional repositories to identify a specific version of an article to believe that any other system at the present time offers the advantages of publishing with learned societies.

In my opinion, the question should not be how will society publishers justify their existence in the future, but rather how can they be better at marketing themselves and promoting the valuable work that they continue to do.

When I say that all association publishers are essentially OA publishers, I mean this from the perspective that associations and their corresponding communities are one and the same. In my opinion, the question should not be how will society publishers justify their existence in the future, but rather how can they be better at marketing themselves and promoting the valuable work that they continue to do. Publishing will always have a cost, whether it relates to print publications or publishing information online. In most well-researched articles I've read on OA, all parties generally tend to agree on this. The real question is where is this money best spent and how. As a longtime publisher who has worked for both for-profit and a leading association publisher, I feel strongly that this is where any debate should be focused, and I am confident that the most valuable and well-run professional society publishers will in the long run continue to prove their worth to the scientific community at large.

Scott E. Delman, PUBLISHER

DOI:10.1145/1536616.1536619

Why Invention and Innovation Diverge

My compliments on the article "One Laptop Per Child: Vision vs. Reality" by Kenneth L. Kraemer et al. (June 2009). It is incredibly valuable for the ACM community to understand the profound difference between invention (which OLPC certainly is as both concept and product) and innovation (the widespread adoption of new mores). How and why political, economic, cultural, and sociological factors influence, if not trump, great ideas, concepts, and products is pertinent with OLPC, especially in light of the project's public visibility.

MediaX@Stanford University (like the MIT Media Lab) encourages the study of how technological solutions affect individuals, organizations, and institutions. We especially encourage small research projects, like OLPC, that pursue "grand ideas" through experimental discovery. Last fall, we hosted Kentaro Toyama, lab director of Microsoft Research India, on "Computing for Socio-Economic Development" in which he described work prompted by a student at Stanford's Center for Innovative Education. Toyama's lab bought and placed several hundred XO laptop computers in Bangalore elementary schools, encouraging students to take them home per Nicholas Negroponte's hope of inspiring parental involvement. To his dismay, many of the machines were stolen and put on the black market where they were worth six months of discretionary family income and clearly too much of a temptation.

The lab concluded that a mouse, at $2 each, had useful attributes: worth nothing on the black market without the XO, could have initials carved onto it without affecting its operation, and inexpensive enough for educators to buy. The Microsoft team designed a "mouse docking station" that could accommodate up to 10 mice, color-coding each cursor on screen so students would require far fewer machines. Despite initial worry that the students would be confused by the multiple cursors, experiments found no particular difficulty with this new operating mode.

Learning could now truly begin. Working in classrooms much larger than those in the U.S., Bangalore's teachers are seldom able to help individual students even if they get stuck, though classmates quickly recognize when their fellow students need help and come to their aid. An early discovery with the XO was that students mastered arithmetic in one-third the time and retained vocabulary drills far longer. Research also found that boys, as well as girls, begin to exhibit cooperative rather than competitive behavior in games and problem-solving sessions on the machines.

Microsoft Labs built a simple reference model—MultiPoint, available as a software development kit—that has since been adapted for teachers in the U.S. and anecdotally found to have similar educational value (http://www.microsoft.com/unlimitedpotential/TransformingEducation/MultiPoint.mspx).

MediaX researchers often find analogous dichotomies between designer functionality and the intended user community at a more systemic level than those usually considered by HCI designers. These techniques, coupled with Kraemer et al.'s excellent coverage, provide additional skills and approaches to the ACM design community.

Charles House (past president of ACM), Stanford, CA

Technologists have a moral duty to ensure that their activities contribute to solving the problems at hand and not diminish other, better, solutions. In this light, the analysis by Kenneth L. Kraemer et al. (June 2009) was helpful in articulating some of the dangers that befall technology projects in sub-Saharan Africa where establishing a vibrant education system in rural areas is a wholly different proposition from its counterpart in urban areas. Schools even a few kilometers from a large town have markedly less-developed infrastructure than those in town. The result is that education often must wait until children are old enough to walk those kilometers to the nearest school.

Try to imagine what OLPC project success would look like in such a context. A typical rural school is constructed with great commitment by the local community but consists of only mud walls, tin roof, and muddy floors. It has a thousand students but no running water, electricity, sanitation, or food service or even enough pens and paper. It is staffed by surprisingly dedicated but inadequately trained, underpaid, and undervalued teachers. Now imagine that the same school receives a large stock of laptops (even if specially designed) that promise a pedagogical revolution. I find such a prospect laughably unrealistic.

It was therefore surprising to read that initial OLPC trials should be conducted in Addis Ababa, the capital of Ethiopia, through a large-scale deployment (50,000 XOs), presumably much of it in rural areas. This imposes on the government an unrealistic expectation to establish a technical-support infrastructure, satellite distribution of digital books, and large-scale teacher-training program. This in a country that invests heavily in improving school enrollment and dramatic university-expansion programs but has difficulty ensuring enough textbooks for its children.

All this is in marked contrast to another initiative emanating from MIT. The online open courseware initiative is well known; less well known is the initiative to put open courseware onto hard drives for distribution to eligible educational institutions with poor Internet connectivity. How helpful it would have been if more MIT professors included adequate reading materials in their open courseware offerings.

OLPC appears to give priority to a technocratic solution to what is essentially a social problem. Technology to support pre-service and in-service teacher education is a much more urgent priority. Incremental advances in technology infrastructure must be used to develop technical skills. That way, the development of teacher and support-technician skills would support future possible large-scale computer deployments.

Imagine how different OLPC implementation would be if it were instead conceived as "one laptop per teacher."

Julian M. Bass, Addis Ababa, Ethiopia

Refocus Fragmented CS Conference Culture

Moshe Y. Vardi was brave (and absolutely right) in taking on CS conference culture in his Editor's Letter "Conferences vs. Journals in Computing Research" (May 2009). Consider how a new technique develops: Prof. Fragment has an idea and publishes it at a workshop. Several of his students refine and modify it over the next few years, publishing their work in various conference proceedings. Another of his students identifies a crucial application of the work. Finally, Prof. Fragment gives an invited lecture at a major conference, covering the full story supplemented with compelling empirical evidence. Unfortunately, the lecture is not accompanied by a paper. Now everyone wants to use the powerful new technique, asking Prof. Fragment for permission to spend a sabbatical term with him at Jigsaw University. He is able to accommodate only one old pal, while everyone else makes do with his group's five conference papers and the online slides of the invited talk. This typically consists of more than 100 pages of material with overlaps, omissions, and contradictions, because each paper represents a different stage of the overall work.

Prof. Fragment would do the CS community a service by publishing his technique in one coherent journal article, presenting the method, together with refinements, applications, and empirical results. But if he were to write such a paper, some referees would likely recommend rejection on the grounds it contained nothing new.

Lawrence C. Paulson, Cambridge, England

Begin with an Author's Response to Reviews

We agree with Ken Birman's and Fred B. Schneider's Viewpoint "Program Committee Overload in Systems" (May 2009) regarding the shortcomings of the conference-review system. A key factor is the lack of incentives for authors to submit their best possible papers, leading, as Birman and Schneider described it, to yet more submissions, along with larger program committees to accommodate them, overworked volunteer program-committee members, and less-informed decisions, as these members are able to read only a small percentage of all submissions.

To discourage repeated submission of the same manuscript—often unchanged—to many different conferences, we propose a simple solution: require that all conference submissions provide two things:

History. A review history of the submission that includes the previous venues to which it was submitted, along with the submitted versions and the reviews the authors received; and

Summary. An explanation of how the authors addressed the concerns cited in previous reviews.

This approach would allow program committees to build on the expert knowledge of the previous review committees where the level of expertise might have differed from their own. Broadly used, it would reinforce the argument that publishing in a CS conference is like a journal publication in other disciplines. It also means authors are further motivated to address or rebut the concerns cited in previous reviews, resulting in a better understanding of the concerns. Finally, authors would be discouraged from "shopping the paper around" until it meets the bar of a particular program committee, encouraging them instead to prepare an acceptable manuscript before its initial submission.

Though this requires extra work by authors at submission time, that work is worth the benefit ultimately received. The concern that some authors might not disclose previous submissions can be addressed the same way simultaneous submissions are treated—by clearly stating the policy and consequences for violating this trust.

Changing the existing system is not difficult. All it takes is one program-committee chair doing an experiment. If it's a good idea and works, others will follow. If not, it will get everyone thinking about alternatives. Using an author's response to reviews is a recent innovation but is already widely used in the CS community.

Jose Nelson Amaral,
Edmonton, Alberta, Canada
Michael Hind, Yorktown Heights, NY

Name the Field: Computer or Computing

The words we use to describe things profoundly affect how we think about them, and once a term has gained a foothold in a field, changing it and its effects is nearly impossible. An example is Peter J. Denning asking "What is computer science?" in his Viewpoint "Beyond Computational Thinking" (June 2009). It has always seemed strange to me that we call our field "comput*er* science" rather than "comput*ing* science," focusing our attention on the hardware and its design and programming rather than on the higher-level great principles that are in fact independent of hardware and historically part of other scientific disciplines.

Perhaps we should change our titles and business cards to read "computing scientist" from "computer scientist" to emphasize this point.

Harry J. Foxwell, McLean, VA

Author's Response:

I endorse Foxwell's view of "computing science." Old names stick when they establish a brand that people generally like. That seems to be the case for "computer science," a name established by the founders of the field in the 1950s. At that time, people in computationally intensive application areas (such as physics and statistics) called what they did "computing science," but the computer scientists of the day felt that "computing" represented applications and not the central focus of R&D. In the 1980s, we began using the term "computing" in place of "computer science and engineering." Twice in ACM history some members proposed calling the organization the Association for Computing, but the required constitutional amendments failed, probably because the voting members liked the long-established brand name. Today, ACM goes by the motto "Advancing Computing as a Science and Profession." We're getting there.

Peter J. Denning, Monterey, CA

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.

DOI:10.1145/1536616.1536651

In the Virtual Extension

Communications' *Virtual Extension brings more quality articles to ACM members. These articles are now available in the ACM Digital Library.*

What Determines IT Spending Priorities?

Hoon S. Cha, David E. Pingry, and Matt E. Thatcher

This article examines findings from a survey of 1,495 business leaders to determine IT spending priorities across business functions and the impact of firm and industry characteristics on these priorities. Survey results show the respondents' highest IT spending priorities are in the areas of administration and production and distribution while the lowest priorities are in research and development and security. In addition, factors such as industry type, firm size, and perceptions of the impact of past IT investments on product quality and revenue affect the allocation of IT expenditures across business functions.

Distinguishing Citation Quality for Journal Impact Assessment

Andrew Lim, Hong Ma, Qi Wen, Zhou Xu, and Brenda Cheang

Having a scientific way to evaluate journal influence based on cross-citations is important for researchers in order to identify impactful journals to submit their work. Earlier literature showed that citation analyses were determined largely through citation quantity. This article presents a novel approach that enables the reflection of citation quality as well. The authors applied this approach to 27 computing journals in the IS field to evaluate the resulting influences on both technical and social technical communities. To facilitate future research, they created a Web application system that allows influence analysis for over 7,000 journals.

Attracting Native Americans to Computing

Roli Varma

Based on empirical data collected by the author, this article discusses dichotomy between economic and socio-cultural factors for Native Americans to pursue education in computer science. It shows cultural, social, and economic factors juxtapose and complement each other, and one without the other would not be adequate to explain the challenges Native Americans face in CS education.

The Critical Elements of the Patch Management Process

Thomas Gerace and Huseyin Cavusoglu

Only a few years ago the term "patch management" was not part of the vernacular in the most advanced IT staffs. Today it is one of the more essential responsibilities of IT departments. The possibility of security threats can decrease by systematically applying patches to software products for which vulnerabilities have been identified. The success of the patch management process depends on several critical elements. This article explores the results of a survey of IT professionals to determine the importance of these critical elements in the patch management process.

Learning to Build an IT Innovation Platform

Rajiv Kohli and Nigel P. Melville

Innovation is a path for successfully competing in free markets, and a firm's IT platform is a key enabler. Organizations that can adapt to changes in the marketplace will continue to thrive and innovative. A multicompany case study finds that three faces of adaptation—customers, people, and creativity—and processes combine to form an IT innovation platform upon which successful companies create new sources of growth. Through six lessons for managers the authors provide practical guidelines on how companies can prepare to build an IT innovation platform to exploit people's creativity, integrate information to identify innovation opportunities, and deliver novel products and services.

Technical Opinion: What Drives the Adoption of Antiphishing Measures by Hong Kong Banks?

Inranil Bose and Alvin Chung Man Leung

Hong Kong has been a hotspot of phishing attacks and since majority of these incidences occurring worldwide are related to the financial services industry, banks in Hong Kong have been frequent targets. The authors studied Hong Kong banks in 2005 and 2007 to assess their phishing readiness and to understand the driving forces that shaped their adoption of anti-phishing measures. They discovered that banks that had smaller assets, higher number of online customers, or frequent attacks tended to be better prepared against phishing.

Ranking Billions of Web Pages Using Diodes

Rohit Kaul, Yeogirl Yun, and Seong-Gon Kim

The most fundamental task of search engines is to rank a large number of Web pages according to their overall quality, without yielding to the relentless attempts to manipulate their rankings. Conventional ranking algorithms based on link analysis have fundamental limitations exploited by many new types of spamming techniques. The authors show a new ranking method using an equivalent electronic circuit to model the Web with diodes in place of hyperlinks produces not only more intuitive and objective rankings than conventional link analysis, but also more effective measures against sophisticated search engine spamming techniques.

Global Software Development: Where are the Benefits?

Eoin O Conchuir, Pär J. Ågerfalk, Helena H. Olsson, and Brian Fitzgerald

Global Software Development (GSD) is gathering great interest in the software industry, as companies seek to realize such benefits as: reduction of costs; 'follow-the-sun' development model; potential access to a larger developer skill-base; potential for increased innovation and transfer of best practices; and closer proximity to customer markets. However, many challenges arise in GSD relating to communication, coordination, and control of the development process. Consequently, much research and effort has attempted to overcome these challenges, and potential benefits are taken for granted as realizable. The article suggests a definite mismatch in the extent to which benefits are realized in practice.

Emerging Software Technologies

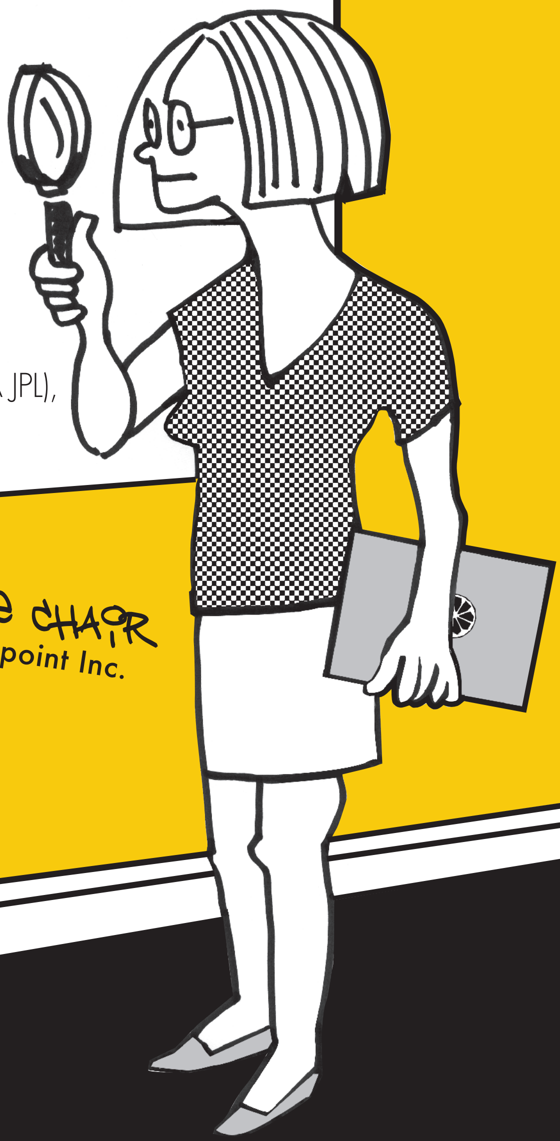
OOPSLA

ORLANDO 2009

Seeding the Clouds
The New World of Software Development
Software Engineering Tools: The Next Generation

Disney's Contemporary Resort, Orlando, FL
October 25-29 2009

Keynotes & Invited Speakers:
Barbara Liskov (Turing Award), Tom Malone (MIT), Jeanette Wing (CMU, NSF), Gerard J. Holzman (NASA JPL), Robert Johnson (Facebook), Jimmy Wales (Wikipedia)



PROGRAM CHAIR
Gary T. Leavens
University of Central Florida
papers@oopsla.org

CONFERENCE CHAIR
Shail Arora, Gradepoint Inc.
chair@oopsla.org

ONWARD! CHAIR
Bernd Brügge
Technische Universität München
chair@onward-conference.org

Association for Computing Machinery
Advancing Computing as a Science & Profession

OOPSLA is sponsored by ACM SIGPLAN in cooperation with SIGSOFT

www.oopsla.org

BLOG@CACM

The *Communications* Web site, http://cacm.acm.org, features 13 bloggers in the BLOG@CACM community. In each issue of *Communications*, we'll publish excerpts from selected posts, plus readers' comments.

DOI:10.1145/1536616.1536620 http://cacm.acm.org/blogs/blog-cacm

An ICT Research Agenda, HPC and Innovation, and Why Only the Developed World Lacks Women in Computing

Jeannette M. Wing writes about the need for a comprehensive research agenda, Daniel Reed discusses high-performance computing, and Mark Guzdial shares insights about women in computing.

From Jeannette M. Wing's "Windmills in the Water"

Windmills in the water were my first sight during my approach to Kastrup, Copenhagen's airport, flying in from Zürich. Blades gracefully spinning in the air—a surprisingly serene sight. Wind power supplies 20% of Denmark's power grid, with the goal of 50% by 2025. Denmark, a country of five million, is itself an experiment in alternative energy.

Apropos, I was on my way to Helsingør, known as the home of Hamlet's castle, to attend an Organization for Economic Cooperation and Development (OECD) conference on Information and Communication Technologies (ICTs), the environment, and climate change. OECD's mission is to help governments tackle questions that affect the global economy, society, and policy. The conference was heavily populated by high-level government officials and industry executives; I was one of a handful of academics present.

At the opening plenary roundtable, I posed this question to the audience: What are the scientific and technical challenges that the ICT research community should be working on today, in anticipation of tomorrow's energy and environment problems? The reason I wanted government and industry officials to hear the word "research" is because I sense that nonscientists might think it's a mere matter of money and a mere matter of the deployment of existing ICT technology to solve these problems. I don't think so.

ICTs account for 2% of global carbon emissions, according to estimates by Gartner. So, reducing our footprint with more energy-efficient devices, computers, and data centers will have a direct effect on ICTs' carbon footprint. Moreover, we need to look at the entire product lifecycle. And what about ICTs' role in the other 98%? ICTs will enable smart cars, smart buildings, smart infrastructure, smart grids, and smart logistics; they will enable telecommuting, telepresence, and telemedicine. So, ICTs also have an indirect effect by helping other sectors save energy. Finally, what about systemic effects? First, algorithms, software, computational methods, computers, and networks are foundational to sensing, modeling, and simulation; used by engineers for building smart things; and used by scientists to observe and model the environment and climate; so, our science and technology will help others attain their sustainability goals. Second, ICTs are just part of a much larger system of systems: it's the interactions and the nonlinear coupling effects of energy, the environment, and the economy that need to be modeled and understood (again, with help from ICTs).

With my question, I raised the attention of government and industry leaders at the conference to the importance of research and the role of academia in the academia-government-industry ecosystem. On the other hand, I have not seen any formal study by or for the research community that frames a research agenda for ICTs and their role in energy, the environment, climate science, or, more broadly, sustainability. This blog entry calls for the diverse readership of *CACM* to spark a discussion on what a comprehensive research agenda might look like.

From Daniel Reed's "HPC: Making a Small Fortune"

There is an old joke in the high-performance computing (HPC) community that begins with a question, "How do you make a small fortune in high-performance computing?" There are several variations on the joke, but they all end with the same punch line: "Start with a large fortune and ship at least one generation of product. You will be left with a small fortune." Forty years of experience, with companies large and small, has confirmed the sad truth of this statement.

As we all know, the computing industry is extremely competitive, and new trends and technologies have repeatedly had a transformative effect. One need look no further than the regular inductees to the Dead Supercomputer Society to see the devastating effects of the ongoing attack of the killer micros on the market for custom HPC system designs. The microprocessor performance increases over the past 30 years due to decreasing feature sizes, higher clock rates, and greater architectural complexity have repeatedly dashed the hopes of many HPC entrepreneurs.

The market lesson is that one false step inevitably leads to failure, particularly for startup companies struggling to establish a new niche in the face of commodity economics. It has never been truer than in today's economy in which potential buyers are retrenching and evaluating each purchase with a discriminating and sometimes jaundiced eye. Recently, the HPC industry lost several established companies to merger and acquisition, due to weak market positions. We have also seen startup companies fail due to missteps and financial pressures.

This reminds me of another old analogy, which compares building computer hardware and software to playing pinball—one's reward for playing well is the opportunity to keep playing via free games. The punishment for not playing well is equally clear; one must continue to insert quarters into the machine.

Without a doubt, we need a new generation of HPC systems, from consumer devices to exascale platforms, to drive innovation, improve health care, manage critical infrastructure, and ensure national safety and defense. The question is whether the rise of multicore and manycore chips and explicit parallelism in the commodity microprocessor and graphics processing unit markets will finally change a few of the rules of the pinball game.

I believe we are at an inflection point, where new approaches must both survive and flourish if we are to continue to deliver higher performance in effective and reasonable ways.

We cannot be complacent about the future, especially now. We must continue to innovate, even if—especially if—that means inserting quarters in the innovation machine.

From Mark Guzdial's "Only the Developed World Lacks Women in Computing"

The National Center for Women in Information Technology meeting at the Googleplex was probably my favorite of its meetings yet. The Academic Alliance meetings were very focused and productive, but what really knocked it out of the park for me were the great talks on cross-national studies of women in IT.

Vivian Lagesen of the Norwegian University of Science and Technology presented her study of Malaysia, where the 52% of all CS undergraduate majors are female. Vivian interviewed students, department chairs (mostly female), and a dean (female). She found that Malaysians can't understand why anyone would think computing is particularly male—if anything, they consider it more female, since it's safe, mostly inside work "like cooking." Vivian found that the three primary influences on students going into CS were their personal enthusiasm, parental interests and wishes, and job prospects, with the last two being much more important than the first. And she concluded that the gendering of computing is constructed by the West, not at all inherent to the field.

Maria Charles of the University of California, Santa Barbara presented her take on the problem, using multinational studies. She says the problem of gender inequality is due to a belief that genders are "different but equal," and that members of different genders are so different that they might as well be from different planets. She thinks that making claims that "CS has characteristics *X* and *Y* that will attract women" only serves to highlight essentially false differences between the genders. Differences in attitudes about math and sciences between men and women are greater in the developed world than in the developing world, where women and men see math and science pretty similarly. In the developing world, computing (and math and sciences) is a great career choice, and that's what drives interest. In the developed world, women make education and career choices as a form of self-expression, so they opt out of science, technology, engineering, and mathematics (STEM) fields early. She suggests that forcing all students to take more STEM classes would give them the opportunity to discover their interest and aptitude for those fields.

My approach to getting more diversity in our computing classrooms is to make the curriculum more relevant to students. An argument I get is, "We're teaching essentially the same topics in the same way today as we did when there were more women in computing. How could the introductory curriculum matter? And if all introductory classes meet the same ACM/IEEE standards, how could the curriculum lead to differences in one part of the world than another?" I think these studies point out that students today are different, they have different goals, and developed world students are looking for something different than students in Malaysia or India. It then makes sense to do something different, if we want a different result. C

Jeannette M. Wing is a professor at Carnegie Mellon University. **Daniel Reed** is vice president of the extreme computing group at Microsoft Research. **Mark Guzdial** is a professor at the Georgia Institute of Technology.

DOI:10.1145/1536616.1536621

David Roman

The New Searchers

Behind Google's simple search box are a complex set of algorithms. Search experts say they are updated constantly, but the same old search page and the listed results don't hint at the work being done to improve search. That work can be tracked on the *Communications* Web site, which uses the Google Search Appliance.

Google executives discuss a major change in the company's approach to search in an interview series by *Digital Daily's* John Paczkowski posted under the Opinion bar on *Communications*' site: http://cacm.acm.org/opinion/interviews/30077-google-and-the-evolution-of-search-whats-next-in-search-much-much-better-search/fulltext. The company now uses an unspecified number of individuals called Quality Raters located around the world to evaluate and improve results. They've added a human touch to the search process. That's just the start.

Researchers are looking to apply spectral graph theory to improve Google's PageRank algorithms, as reported by Kirk L. Kroeker (http://cacm.acm.org/magazines/2008/9/5305-finding-diamonds-in-the-rough/fulltext), and Google is trying to index and add Deep Web pages to the billions it searches today, notes Alex Wright (http://cacm.acm.org/magazines/2008/10/535-searching-the-deep-web/fulltext).

The unchanged Google page doesn't hint at any of this, but the changes are evident to experts when they plug the same terms into Google and perform a search repeatedly over time. When the search is over the results are different.

A more contemporary face of search is Cornell Professor Jon Kleinberg, whose efforts to determine relevant, trusted sources were embedded into the Hubs and Authorities algorithm. Kleinberg was awarded the 2008 ACM-Infosys Foundation Award in the Computing Sciences, cited for his contributions to improving search techniques employed by billions of users worldwide (http://cacm.acm.org/news/25188-network-pioneer-cited-for-revolutionary-advances-in-web-search-techniques/fulltext). Searchers' tendency to click the top item on a search page reinforces the primacy of sources identified by Kleinberg's linkage-based algorithms, which may prompt further refinement (http://cacm.acm.org/magazines/2008/2/5454-are-people-biased-in-their-use-of-search-engines/fulltext).

Kleinberg lecturing at Cornell.

ACM Member News

HABERMANN AND WILLIAMS WIN SIGSOFT INFLUENTIAL EDUCATOR AWARD

The late A. Nico Habermann, founding dean of Carnegie Mellon University's School of Computer Science, and Laurie Williams (left), associate professor of computer science at North Carolina State University, were honored by SIGSOFT with its inaugural Influential Educator Award.

Habermann was cited "for significant and lasting contributions to the field of software engineering as a teacher and mentor." His widow, Marta Habermann, accepted the award from William Griswold, SIGSOFT chair, at the International Conference on Software Engineering, in May.

In recognizing Williams for her significant contributions to software engineering and computer science education, particular notice was paid to her outstanding work in propagating pair programming as a widely adopted educational practice, her work in establishing an open seminar environment for software engineering, and in creating an agile software engineering curriculum.

The SIGSOFT Influential Educator Award will be presented annually to an educator who has made significant contributions to, and impact on, the field of software engineering with his or her accomplishments as a teacher, mentor, researcher (in education or learning), author, and/or policymaker.

SIGCOMM 2009

The premier conference on data communications, SIGCOMM 2009 will be held in Barcelona, Spain from August 17–21, 2009. The conference will include sessions on wireless networking, data center network design, novel aspects to networking, and other topics. Jon Crowcroft, a professor of computer science at the University of Cambridge and winner of the 2009 SIGCOMM Award, is the keynote speaker. For more info, visit http://conferences.sigcomm.org/sigcomm/2009/.

KLEINBERG PHOTOGRAPH BY MARC SMITH

Science | DOI:10.1145/1536616.1536622 Don Monroe

Just For You

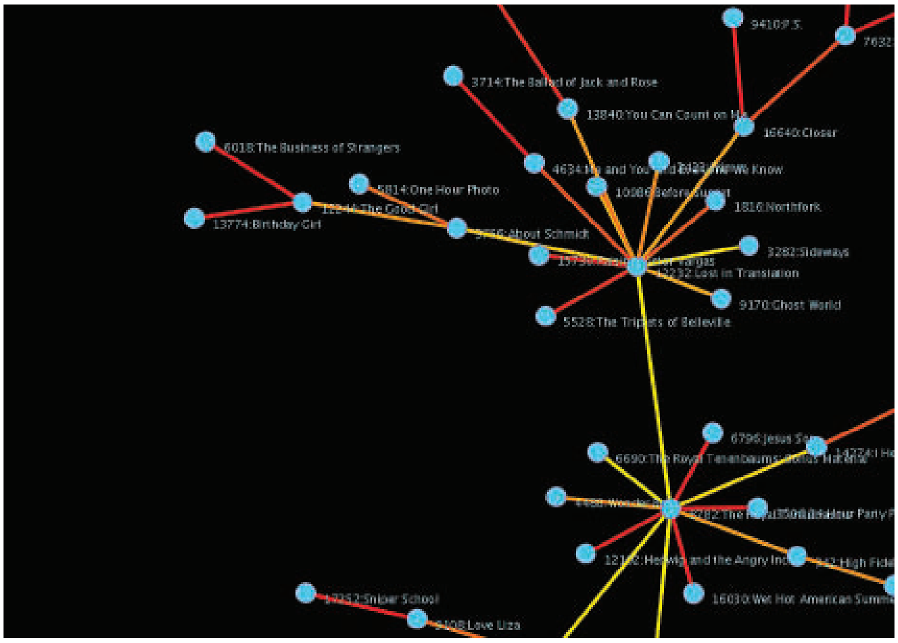
Recommender systems that provide consumers with customized options have redefined e-commerce, and are spreading to other fields.

SOMETIME SOON, A team of programmers is likely to receive a check from Netflix for $1 million. More than 4,000 teams have entered the movie-rental company's Netflix Prize competition, which was established in 2006 to improve the recommender system Netflix uses to suggest movies to its 10 million-plus customers. As this article went to press, a coalition of previously competing teams, calling itself BellKor's Pragmatic Chaos, had edged past the 10% ratings improvement over Netflix's system, which will win them the Netflix Prize (unless another team beats their 10.5% improvement by late July).

The term "recommender system" has largely supplanted the older phrase "collaborative filtering." These systems create recommendations tailored to individual users rather than universal recommendations for, well, everyone. In addition to movie recommendations like those from Netflix, many consumer-oriented Web sites, such as Amazon and eBay, use recommender systems to boost their sales. Recommender systems also underlie many less overtly commercial sites, such as those providing music or news. But in each case a recommender system tries to discern a user's likely preferences from a frustratingly small data set about that user.

One lure of the Netflix Prize for researchers is Netflix's database of more than 100 million movie ratings—which include user, movie, date of rating, rating—from some 480,000 users about nearly 18,000 movies. After training their algorithms with this data, teams predict the ratings for a secret batch of 2.8 million triplets (user, movie, date of rating). Netflix then compares their accuracy to that of its original Cinematch algorithm. Sharing a massive, real-life data set has energized research on recommender systems, says Bob Bell, a principal member of the technical staff at AT&T Research, and a member of BellKor's Pragmatic Chaos. "It's led to really big breakthroughs in the field," says Bell.

From the start, the Netflix Prize has appealed to academically oriented researchers. The eventual winners, as well as the annual progress prize winners (who receive $50,000), agree to publicly share their algorithms, and many teams openly discuss their research in the online Netflix Prize Forum. For researchers, this openness adds to the


9410:P.S.
3714:The Ballad of Jack and Rose
13840:You Can Count on Me
16640:Closer
6018:The Business of Strangers
4634:Me and You and Everyone We Know
5814:One Hour Photo
10086:Before Sunset
1816:Northfork
13774:Birthday Girl
12244:The Good Girl
About Schmidt
3282:Sideways
12232:Lost in Translation
9170:Ghost World
5528:The Triplets of Belleville
6796:Jesus Son
6690:The Royal Tenenbaums
12102:Hedwig and the Angry Inch
High Fidelity
16030:Wet Hot American Summer
Sniper School
Love Liza

Clusters of movies discovered by a computer algorithm created for the Netflix Prize competition, with lines closer to yellow representing stronger similarities and colors closer to red representing weaker similarities.

VISUALIZATION BY CHRISTOPHER HEFELE

intellectual excitement. "People like us are motivated by the research, not necessarily by the money," says Chris Volinsky, executive director of statistics research at AT&T Research and a member of BellKor's Pragmatic Chaos. "Having an academic flavor to the competition has really helped it to sustain energy for two-and-a-half years."

Although Netflix is unique in publicly enlisting and rewarding outside researchers, many other companies are fine-tuning the choices their recommender systems present to customers. Some of their efforts, like those of Amazon, L.L. Bean, and iTunes, are obvious to users. Other companies work behind the scenes, quietly monitoring and personalizing the experience of each user. But either way, user satisfaction depends on not just new and improved algorithms, but individual human preferences, with all of their many quirks.

The Netflix Prize has brought a lot of attention to the field, notes John Riedl, a professor of computer science at the University of Minnesota. However, Riedl worries that the Netflix Prize puts "a little too much of the focus on the algorithmic side of the things, whereas I think the real action is going to happen in how you build interfaces … that expose the information in more creative and interesting ways."

Implicit and Explicit Information

To entice its customers to rate movies, Netflix promises to show them other movies they will enjoy. Netflix also encourages its customers to provide detailed information about their viewing preferences. Unfortunately, this rich, explicit feedback demands a level of user effort that most Web sites can't hope for.

Instead, many companies rely on implicit information about customer preferences, such as their purchasing history. However they get the feedback, though, researchers must manage with a sparse data set that reveals little of many customers' tastes about most products. A further, critical challenge for Web-based recommender systems is generating accurate results in less than a second. To maintain a rapid response as databases grow, researchers must continually trade off effectiveness for speed.

One popular and efficient set of methods, called *k* nearest neighbors, searches for a handful of other customers (*k* of them) who have chosen the same items as the current customer. The system then recommends other items chosen by these "neighbors."

In contrast, latent factor methods search customers' choices for patterns that can explain them. Some factors have obvious interpretations, such as a user's preference for horror films, while other statistically important factors have no obvious interpretation. One advantage of latent-factor methods is they can provide recommendations for a new product that has yet to generate much consumer data.

These algorithms all aim to solve the generic problem of correlating preferences without invoking knowledge of the domain they refer to, such as clothing, movies, and music. In principle, notes Joseph Konstan, a professor of computer science and engineering at the University of Minnesota, as long as individuals' preferences remain constant, then with enough opinions from a sufficient number of people, "you don't need to know anything about the domain." In practice, Konstan says, limited data and changing tastes can make domain-specific approaches more effective.

One of the most sophisticated domain-specific approaches is used by Internet-radio company Pandora, which employs dozens of trained musicologists to rate songs on hundreds of attributes. "We are of the opinion that to make a good music recommendation system you need to understand both the music and the listeners," says Pandora Chief Operating Officer Etienne Handman. Still, the most enjoyed Pandora playlists, he says, supplement the musicologists' sophisticated ratings with statistical information from users.

Measuring Effectiveness

To win the Netflix Prize, a team must beat Cinematch by 10% on a purely statistical measure, the root mean square error, of the differences between predicted and actual ratings. Like content-based assessments, however, this ob-

Obituary

Rajeev Motwani, Google Mentor, Dies at 47

Rajeev Motwani, a professor of computer science at Stanford University who mentored many students and Silicon Valley entrepreneurs, including the founders of Google, died in an apparent accidental drowning on June 5. He was 47.

Motwani was well known for his theoretical research on randomized algorithms and his contributions in data mining, Web search, information retrieval, streaming databases, and robotics. He is the author of two classic computer science textbooks, *Randomized Algorithms*, with Prabhakar Raghavan, and *Introduction to Automata Theory, Languages and Computation*, with John Hopcroft and Jeffrey Ullman.

Motwani served as director of graduate studies in the computer science department at Stanford and founded the Mining Data at Stanford (MIDAS) project. He was known as a friendly, well-respected professor who always made himself available to advise and mentor young entrepreneurs and students, including Google cofounders Sergey Brin and Larry Page when they were graduate students at Stanford in the mid-1990s.

"The Google founders used both his technical expertise and his understanding of how technology can transition into the real world. He was helpful in that regard," recalls Stanford computer science professor Balaji Prabhakar, a friend of Motwani. "Many former students and Silicon Valley folks have sought out Rajeev's advice and input. He was a generous person who saw the potential in people and their ideas."

Motwani was also an angel investor and technical advisor for many startup companies in Silicon Valley, and sat on the board of numerous companies, including Google, Kaboodle, and Mimosa Systems.

Motwani's research earned him numerous awards, including the Gödel Prize and an Arthur P. Sloan Foundation Research Fellowship.

—Wylie Wong

PHOTOGRAPH COURTESY OF STANFORD NEWS SERVICE jective metric falls short of what users want in recommendations.

"The predicted enjoyment [by this measure] is just one factor that goes into supporting the movies that we present," says Jon Sanders, director of recommendation systems at Netflix. The company augments this metric with other features, such as movie genre, which affect the appeal of a movie to a user. In addition, Sanders notes, the Netflix recommendation interface says why a movie was recommended, to build trust that it is selected specifically for that user. "There's much more to personalization than what the Netflix Prize reveals," he says.

Giving users realistic expectations can defuse the mistrust caused by occasional bizarre recommendations (which make for good stories but bad business). On the other extreme, however, conservative suggestions can seem trivial. "It's fairly easy to make recommenders that never do a stupid recommendation," says Riedl. "You actually want to tune the algorithm where it's more likely to make errors," because then "it's also more likely to make serendipitous relationships."

In the case of news, presenting some unexpected connections has societal importance, because readers often gravitate to Web sites that reinforce their beliefs. "There's a big debate in personalization in news in particular, about whether personalization will lead to pigeonholing, like whether people will only read the news that they like," says Greg Linden, who ran a personalized news site, Findory, from 2004 to 2007. Diversity is also critical in other domains. "The key thing with recommender systems is they're trying to help with discovery," Linden notes, unlike search engines that "help you find something you already know you want."

Widening Impact

Recommender systems haven't helped solve the business challenge of earning significant revenue from personalizing the news, but they have transformed traditional retailing. Michel Wedel, a professor of consumer science at the University of Maryland, notes that recommender systems have become "more or less the backbone of many of the major firms on the Web," and the Netflix Prize's $1 million reward hints at the scale of the business they expect to receive. However, Wedel suggests that the best recommender systems are moving away from the explicit ratings used by Netflix, in part because others can intentionally skew the ratings.

In the future, recommender systems might help people navigate everything from work tasks to social relationships, says John Riedl.

The next generation of recommender systems will rely more on implicit information, such as the items that a user clicks on while navigating a site, says Francisco Martin, chief executive officer of Strands, Inc., a recommendation and personalization technologies company in Corvallis, OR. "Based on your navigation patterns, you're correlating products, and you're giving very valuable information to the recommender system," he says. Improved recommender systems also track changing tastes and context-specific preferences. In the not-too-distant future, Martin also envisions bank-based systems that track all of an individual's spending and use that information to make personal finance recommendations. "All of our life will be digitized," he says.

Minnesota's Riedl also imagines applications far beyond commerce, helping people navigate everything from work tasks to social relationships. Many observers have noted that human's biological evolution has been largely overtaken by cultural evolution, he says. But by combining computer and human strengths, Riedl says, recommender systems let us "create systems that take us to new places, new geniuses." C

Don Monroe is a science and technology writer based in Murray Hill, NJ. Yehuda Koren, Yahoo! Research (and a member of BellKor's Pragmatic Chaos), contributed to the development of this article.

Milestones

Computer Science Awards

The American Association for Artificial Intelligence, the Computer History Museum, and the Electronic Design Automation Conference recently honored members of the computer science community for their distinguished achievements.

MARIE R. PISTILLI AWARD

Telle Whitney, president and CEO of the Anita Borg Institute, is the recipient of the tenth annual Marie R. Pistilli Women in Electronic Design Automation Achievement Award. The award honors Whitney for her contri- butions to women working in technology as a role model and as leader of Anita Borg Institute.

AAAI FELLOWS

American Association for Artificial Intelligence's newly elected Fellows are Wolfram Burgard, Albert-Ludwigs-Universität Freiburg; William W. Cohen, Carnegie Mellon University; Andrew K. McCallum, University of Massachusetts, Amherst; Jeffrey S. Rosenschein, The Hebrew University of Jerusalem; Dan Roth, University of Illinois at Urbana-Champaign; Daniela Rus, MIT Computer Science and Artificial Intelligence Laboratory; Robert E. Schapire, Princeton University; Venkatramanan Siva Subrahmanian, University of Maryland, College Park; and Pascal R. Van Hentenryck, Brown University.

CHM FELLOWS

The Computer History Museum's 2009 Fellows are Robert R. Everett, for his work on the MIT Whirlwind and SAGE computer systems and a lifetime of directing advanced research and development projects; Don Chamberlin, for his work on Structured Query Language and database architectures; and the team of Federico Faggin, Marcian Edward "Ted" Hoff, Stanley Mazor, and Masatoshi Shima, for their work on the Intel 4004, the world's first commercial microprocessor.

Technology | DOI:10.1145/1536616.1536623 Kirk L. Kroeker

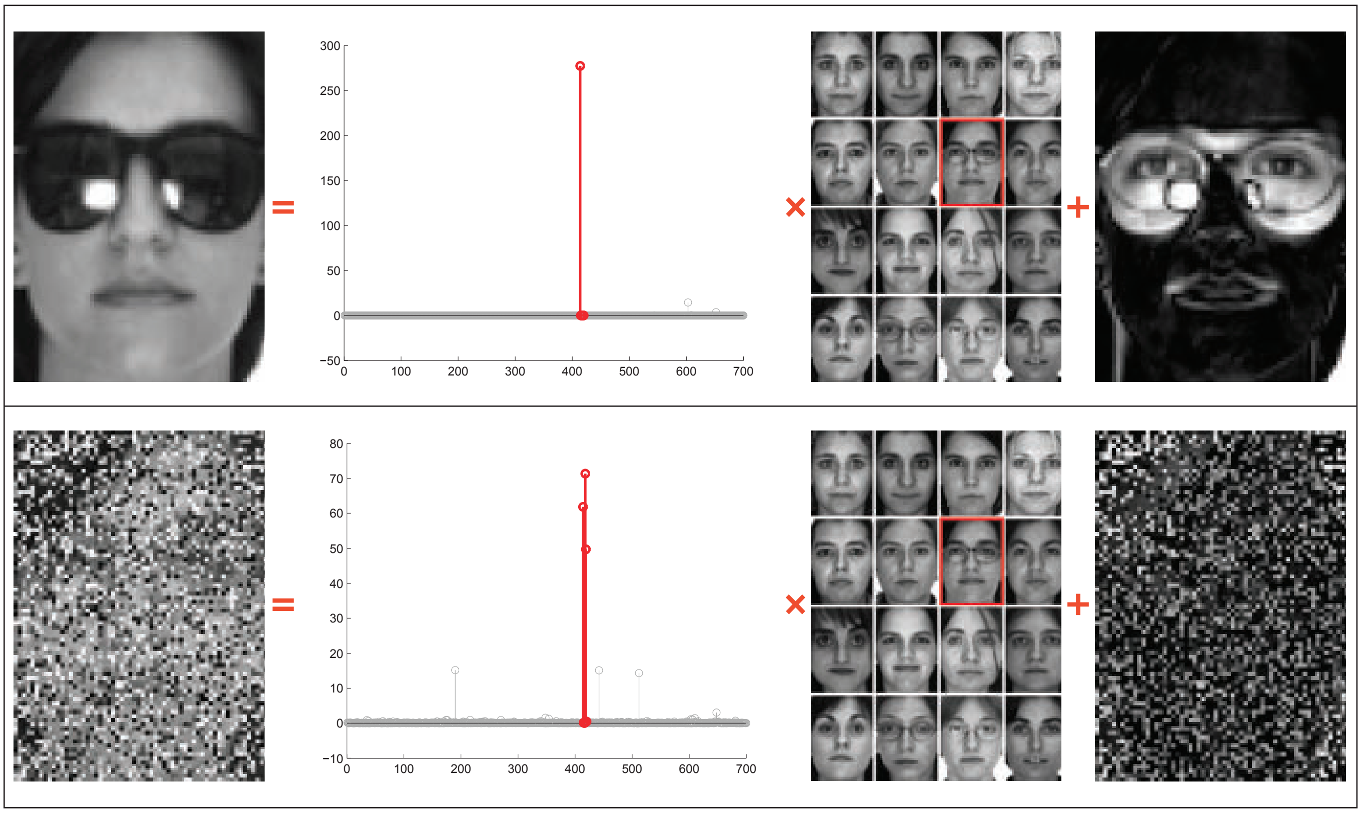
Face Recognition Breakthrough

By using sparse representation and compressed sensing, researchers have been able to demonstrate significant improvements in accuracy over traditional face-recognition techniques.

THE THEORIES OF sparse representation and compressed sensing have emerged in recent years as powerful methods for efficiently processing data in unorthodox ways. One of the areas where these theories are having a major impact today is in computer vision. In particular, the theories have given new life to the field of face recognition, which has seen only incremental increases in accuracy and efficiency in the past few decades. Now, thanks to the application of these theories to classic face-recognition problems, researchers at the University of Illinois at Urbana-Champaign (UIUC) have been able to demonstrate significant improvements in accuracy over traditional techniques.

The idea of applying sparse representation and compressed sensing to face recognition was so novel in 2007 that two papers outlining the method were rejected by mainstream vision conferences. "Just like compressed sensing, our approach to face recognition is completely unorthodox," says Yi Ma, a professor of electrical and computer engineering at UIUC. "The reviewers simply did not believe such good results were possible, or that sparsity was even relevant."

Ma had been studying the sparse representation and compressed sensing theories of Emmanuel Candes and David Donoho while on sabbatical at the University of California, Berkeley in early 2007. It was then, he says, that he connected the theories to computer vision by applying the ideas to problems associated with one of the most readily available sources of raw data for vision research: face images. "The results were far beyond what I had ever expected or imagined from the beginning," Ma says. "What happened next was the most exciting period of research I have ever had."



The face-recognition method developed at the University of Illinois at Urbana-Champaign. On the left, the test face is partially covered with sunglasses (top) and corrupted (bottom). The face is represented in the middle as a sparse linear combination of the training images plus sparse errors due to occlusion (right top) and corruption (right bottom), with the red coefficients corresponding to the training images of the correctly identified individual.

While traditional face-recognition methods record certain facial measurements, such as the distance between the eyes and the width of the mouth, the method developed by Ma turns the conventional wisdom about such strategies on its head by representing faces as the sparsest linear combination of images in a database. Unlike the traditional methods that focus on low-dimensional features, Ma's technique works directly with high-dimensional images as a whole, focusing on their sparse structures.

In a way, says Ma, the idea is simple. "Given a database of face images, our algorithm simply tries to use the fewest possible images to interpret a query image," he says. The idea is based on the notion of treating the given training data as a large dictionary for representing test images. For each test image, the algorithm seeks the sparsest representation from the main dictionary and from an auxiliary dictionary consisting of the individual pixels.

"The use of the data itself as dictionary is a very interesting concept for certain types of data and applications," says Guillermo Sapiro, whose recent work as a professor in the department of electrical and computer engineering at the University of Minnesota has focused on sparse representation and dictionary learning. "The face-recognition work being carried out by Ma and his colleagues represents a novel take on this problem and a very refreshing one that we are all looking at with a lot of excitement."

Accurate Identification

The aspect of Ma's approach that is arguably generating the most excitement is its ability to identify faces despite the images being corrupted or the faces being partially covered. Traditional face-recognition techniques are susceptible to noise and their accuracy is significantly reduced when parts of a face are covered or obscured, such as with a mask or a disguise. Ma says the algorithm can handle up to 80% random corruption or occlusion of the face image and still reliably recognize a person, making it more capable than even the human brain in its ability to identify a face.

"Our theorem even suggests that the percentage of corruption can approach 100% as the dimension goes to infinity," says Ma. In contrast, the accuracy of traditional face-recognition techniques declines to less than 70% when part of a face is covered or if the query image suffers from noise or poor resolution. Ma's algorithm can withstand such occlusions and corruptions because it does not focus on specific facial details, such as the size of a nose. In Ma's algorithm, the sparsest representation accounts for the parts of the face that are not occluded or corrupted. This capability alone has drawn the attention not only of the media, but also of more than a dozen companies that are interested in licensing the technology.

Yi Ma envisions a future in which a user can capture a person's picture with a mobile phone's camera, submit a query, and receive a match moments later.

"We've been pleasantly surprised by the breadth of interest we've received, both from companies working in traditional application areas of face recognition, such as security and access control, as well as in many less-traditional areas," says John Wright, a UIUC graduate student who won the $30,000 Lemelson-Illinois Student Prize this year for his work creating a prototype application using Ma's algorithm.

However, despite the commercial interest, several challenges remain. "Although the initial idea seemed very natural, it has since taken a lot of mathematical work to justify why it should work so well," says Wright. "It also has taken a lot of engineering work to bring it into the real world." One of the challenges, as with other face-recognition systems, is how to achieve high performance with massive data sets.

The UIUC method is based on a scalable algorithm, but Ma says that if the number of subjects becomes large, running the algorithm in real time on current hardware is challenging. Currently, he says, the algorithm can run in real time on a database consisting of up to 1,000 subjects, making it suitable for use in an access-control system for a small company. However, the UIUC team is working on parallel and graphics processing unit implementations so the system can scale to much larger numbers, possibly even to millions of subjects.

Given the algorithm's accuracy, the implications of scaling it to millions of subjects could be significant, leading to new ways of searching for images, annotating multimedia, and even monitoring crowds. For example, Ma envisions a Web-based service that would allow users to capture a person's picture with a mobile phone's camera, remotely submit a query, and have a match returned moments later. "In my view," says Ma, "this would be a killer app for the emerging parallel, distributed, and cloud initiatives."

Another challenge the UIUC team faces is to extend the method to deal with less controlled training data, particularly for applications such as online photo tagging or image searching. For these applications, the researchers say, it is crucial to understand the minimum amount of training data needed for the algorithm to succeed. But for Ma, this challenge, in particular, is not specific to face recognition. "One of the fundamental problems that we need to address in computer vision," he says, "is how to obtain or learn a good dictionary for the problem at hand."

Still, Ma says he and his team remain confident that for applications where the acquisition of training images can be controlled, an obtainable goal for their technology is real-time, highly accurate face recognition that far exceeds the capabilities of current systems. "I am confident that with sufficient support from the industry and possibly from the government," he says, "we could start to see working face-recognition systems based on this research in the next five to 10 years." C

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

News | DOI:10.1145/1536616.1536625 Tom Geller

IT Drives Policy—and Vice Versa

Technologists discuss government policies affecting broadband, patent reform, privacy—and President Obama's effect on it all.

THE U.S. ECONOMIC stimulus package loomed large over the third annual Tech Policy Summit, attended by more than 300 people at the Marriott San Mateo/San Francisco Airport Hotel from May 11 to 13. With the first day dedicated to a broadband innovation agenda, there was particular focus on the $7.2 billion made available for broadband development through the American Recovery and Reinvestment Act. Broadband issues also pervaded the following two days, especially the problem of the "middle mile" connecting the Internet backbone to rural regions increasingly served by local wireless networks.

However, discussions at Tech Policy Summit '09 also ranged to areas outside of broadband and IT as well, with health care and energy policy receiving prominent attention. Whatever the topic, computer technology's role eventually took the foreground. "The number-one threat to the country's financial stability is the cost of health care," noted Blair Levin, managing director at investment banking firm Stifel Nicolaus and cochair of the technology, innovation, and government reform working group for the Obama-Biden transition team. "The Obama administration is saying that IT is central to its solution."

On that note, several participants reported positive interactions with the Obama administration about technology initiatives. Software developer, entrepreneur, and philanthropist Mitch Kapor said he believes that "the level of knowledge within the [Obama] administration is very broad. I didn't see that in the Bush admin, and if you go back to Clinton, it was isolated around Al Gore. It wasn't a priority to understand and take advantage of technology and innovation." *Wall Street Journal* columnist Walt Mossberg agreed. "Obama's the first president who lives in the digital world," he said. "That's not extraordinary. That's the lifestyle that everybody in this room lives. Al Gore did as well, but he was never the president."

Walt Mossberg (left) interviewing Mitch Kapor at Tech Policy Summit '09 in San Mateo, CA.

According to Natalie Fonseca, principal of conference organizer Sage-Scape, Tech Policy Summit '09 drew slightly more than half its participants from the Silicon Valley region, with the most of the remainder coming from Washington. Fonseca, too, believes Silicon Valley is more willing to deal with government now. "There was interest in government involvement before, but now there's more opportunity for such initiatives to be successful," she said. Jim Dempsey, the vice president of public policy for the Center for Democracy and Technology, agreed. "When I came to the first one of these conferences three years ago, it was heavily dominated by D.C. people," he said. "But today, it's a lot more entrepreneurs—and to me that's good. Innovators need to understand and be involved in policy, and not just say, 'Give us more H1-B visas.' "

U.S. Representative Mike Honda (D-CA) was among those who echoed the need for technologists to participate in Washington-based policy decisions. "Years ago I asked technologists, 'What do you want me to do for you?' " he said. "They answered, 'Just stay out of my way, because whenever you guys do anything, it ends up being a lot more work for me.' But as a former schoolteacher, I know that field trips teach people a lot. If policymakers don't know what they're looking at, the policies they make won't make any sense. So it's a really good idea for you to talk to leaders and policymakers." C

Tom Geller is an Oberlin, Ohio-based freelance writer covering science, technology, and business.

PHOTOGRAPH BY ANDREW FEINBERG

Society | DOI:10.1145/1536616.1536624

Leah Hoffmann

Learning Through Games

Electronic games can inspire players to explore new ideas and concepts. By gaining a better understanding of the dynamic between player and game, researchers hope to develop more interesting and effective approaches.

IT WASN'T UNTIL after he'd stopped working as a middle school choir teacher and joined one of Microsoft's software testing units that Jeremy Tate first encountered *Guitar Hero*. A gaming enthusiast, Tate was quickly hooked. And as he grew more familiar with the game and observed others at play, he noticed how *Guitar Hero* helps gamers master challenging musical concepts, such as phrasing and rhythm, notions he had struggled to teach his own students.

"Players are taught instantly, as a function of the game," Tate explains. "Want a better score? Do it right next time." He's now talking to several teachers in his old school district about putting his observations to use and bringing music video games like *Guitar Hero*, *Lips*, and others into the classroom.

Games like ***Guitar Hero III: Legends of Rock*** simulate the playing of a real guitar and teach players how to master challenging musical concepts, such as phrasing and rhythm.

In many ways, of course, it's not surprising that educators could make use of an electronic medium teens have already widely embraced. According to a 2008 survey conducted by the Washington, D.C.-based Pew Research Center, 97% of children between the ages of 12 and 17 play computer, Web, portable, or console games. Among many adults, the popular perception of these games is that they have little redeeming value and may harm children by desensitizing them to violence. Over the past few years, however, a new body of research has begun to demonstrate how games can have a positive effect on youngsters by stimulating their imaginations, sparking their curiosity, and promoting the exploration of difficult issues and concepts. Off-the-shelf games like *Sim City*, *Civilization*, and *Railroad Tycoon* have been successfully used in the classroom to help students understand complex social, historical, and economic processes. Tim Rylands, a teacher at a primary school near Bristol, England, made headlines in 2005 for his award-winning use of *Myst* to improve students' writing skills. And *World of Warcraft* has been praised by educational researchers at the University of Wisconsin-Madison for its ability to foster abstract thinking among middle and high school students, who meet online to share strategies and ideas about the game.

"Games are goal-directed learning spaces," says James Gee, a professor of literacy studies at Arizona State University who has done extensive work on the subject. According to Gee, games give children the tools they need to explore complex systems and experiment with different possibilities and outcomes. Rather than simply memorizing figures and statistics, children learn to constructively use facts to solve problems. In a game whose objective is to design and build a city, for example, kids end up not only learning about building codes, but how to put them to use. In *Civilization III*, a game in which players lead a civilization from 4,000 B.C. to the present, students are frequently motivated to consult maps, Wikipedia, and other external resources to get ahead.

"Could there be a better learning philosophy for the 21st century?" asks Gee.

Games as Interactive Platforms

The tricky part, of course, is figuring out why certain games advance learning. What factors keep students engaged? What features encourage them to apply what they've learned to real problems? Too many titles that are currently marketed as educational games, experts say, are little more than digital flashcards, presenting students with straightforward drills in subjects like math and grammar rather than giving them an interactive platform through which to explore new ideas and concepts. By gaining a more sophisticated understanding of the dynamic between player and game, researchers hope they can de-

PHOTOGRAPH BY WŁODI

velop more interesting and effective approaches.

To meet those challenges, specialized research groups have sprung up at universities across the country. New York University's Games for Learning Institute (G4LI), for example, offers a forum for experts in disciplines like computer science, cognition, and educational research to collaborate on experiments and research. Founded in 2008 with funding from Microsoft, G4LI's mission is to conduct rigorous empirical research into how games can support learning. Thus far, work has focused on science, technology, engineering, and math topics and on middle school, when children typically lose interest in those subjects.

"We're moving from individual case studies that address the effectiveness of a single game to a more descriptive, qualitative type of research," says Jan Plass, a professor of educational communication and technology at NYU and G4LI codirector. "We observe game play, we test things empirically ... we're interested in finding patterns." Plass's colleagues in the computer science department then build mini-games to test effective features—a particular incentive system or type of player support, for example—and further refine their understanding. G4LI researchers have also reached out to game developers and educators to analyze their experiences. The ultimate goal, says Plass, is to develop a comprehensive set of principles and standards that could help people effectively design, build, and use educational games.

Other game-related research is ongoing at University of Wisconsin-Madison, Massachusetts Institute of Technology, and Indiana University. Likewise, Games for Change, a New York nonprofit organization that focuses on social justice issues, provides an additional platform for people to share ideas, resources, and tools. Though many findings are preliminary, one important theme that's emerged is the need for teaching material to be integrated into the framework of a game's design rather than added to it later. "You need something that allows knowledge to unfold," says Katherine Isbister, a professor of digital media and computer science and engineering at New York University's Polytechnic

"Games," says James Gee, "are goal-directed learning spaces."

Institute and an investigator at G4LI. It's therefore important to start with a definite set of ideas and objectives rather than a structure of play.

Ian Bogost, a founding partner of Atlanta-based gaming studio Persuasive Games, concurs. Effective educational games, says Bogost, construct a model of how a particular issue or subject works. Players then interact with that model to understand its contours and reasoning, and can ultimately decide whether to embrace or reject it.

"It's through the experience of making choices that you learn," he asserts.

Bogost and his colleagues build educational games for a diverse set of corporate and nonprofit clients. Large companies want games that give them a more engaging way of training new workers, or make themselves more appealing to younger customers. Political and nonprofit organizations, on the other hand, are trying to reach teens and educate them about problems like climate change and poverty. Persuasive Games has also developed several issues-based games of its own. Unfortunately, Bogost says, it can be difficult to identify a market for these projects. ("If it's online, people expect it to be free," he sighs.)

Integrating games into secondary school classrooms can be challenging. Each district has its own curriculum and objectives. Some teachers are skeptical about gaming's pedagogical value, while others are unfamiliar with the variety of available games. For now, researchers say, the easiest way to get games into the classroom is at a grassroots level, and G4LI and other academic institutes are working hard to foster relationships with local schools.

At the university level, where games enjoy more widespread curricular support and adoption, computer science departments in particular have begun to feel an effect. The most recent Taulbee Report indicates an upsurge in interest and enrollment in the field, and anecdotal evidence suggests that gaming may have played a role as games and gaming techniques help make core computer science principles more accessible to students. The prospect of being able to join the still-growing game development industry has also attracted new prospects to the field.

As researchers continue to work out principles of learning, cognition, and design, they would be well advised to keep an essential principle of the gaming industry in mind: Make it fun. It may not be what most teens think of when they think of school. But as Isbister points out, it's at the heart of why they play games—and is one of the main things that keeps them engaged and willing to persist in ways that many teachers only dream of.

Leah Hoffmann is Brooklyn-based science and technology writer. Ken Perlin, New York University, contributed to the development of this article.

Call for Nominations for CACM GENERAL ELECTION

The ACM Nominating Committee is preparing to nominate candidates for the officers of ACM: President, Vice-President, Secretary/Treasurer; and two Members at Large.

Suggestions for candidates are solicited. Names should be sent by November 5, 2009 to the Nominating Committee Chair, c/o Pat Ryan, Chief Operating Officer, ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701. With each recommendation please include background information and names of individuals the Nominating Committee can contact for additional information if necessary. Stuart Feldman is the Chair of the Nominating Committee.

Government Policy | DOI:10.1145/1536616.1536626

Gregory Goth

U.S. Unveils Cybersecurity Plan

'Intent and timing' may help the federal cyberspace initiative work better than previous blueprints.

THE NATIONAL CYBERSECURITY initiative announced by President Barack Obama last May follows a decade of similar efforts by the two preceding administrations—and after a decade of hearing earnest governmental pronouncements about how vital cybersecurity is, skeptical observers might say little has been accomplished except to demonstrate the intricacies of bureaucratic battles in the creation of new government agencies.

However, crucial differences exist between the Obama administration's cybersecurity efforts, marked by the release of its 60-day *Cyberspace Policy Review* (http://www.whitehouse.gov/assets/documents/Cyberspace_Policy_Review_final.pdf), and those of Bill Clinton and George W. Bush, despite the many similarities, says James Lewis, senior fellow for technology and public policy at the Center for Strategic and International Studies, a Washington, D.C.-based think tank.

"The difference here is in intent and timing," Lewis says. "This administration did this in their first few months in office, and it looks like the president has an interest in it. The Clinton administration's Presidential Decision Directive 63 and the Bush administration's Comprehensive National Cybersecurity Initiative were both done late in their second terms and didn't really get any traction."

Lewis and security experts Fred B. Schneider, professor of computer science at Cornell University, and Susan Landau, distinguished engineer at Sun Microsystems, say the Obama administration must work deftly if its cybersecurity plan will emerge with more credibility than its two predecessors. Among the vital elements they said the Obama administration's report contained was recognition that the federal

President Barack Obama speaking about the U.S. government's *Cyberspace Policy Review* at the White House on May 29, 2009.

PHOTOGRAPH BY MANDEL NGAN / GETTY IMAGES

government must take an active role in operating cybersecurity policy and infrastructure; it must balance that active role with a concerted campaign to protect industry's ability to innovate in the creation of new platforms and applications; it must preserve citizens' confidence that cybersecurity policy will protect their civil liberties as well as the cyberinfrastructure; and it must forge workable partnerships with other nations, nongovernmental organizations, and technical standards bodies.

Landau says perhaps the report's most important indicator of the new administration's cybersecurity strategy is a passage in the report's executive summary in which the phrase "national economic needs" precedes "national security requirements."

The cybersecurity czar must be appointed at a high enough level to possess real clout.

However, the new emphasis adds more interested parties—the cybersecurity czar is expected to report to both the National Economic Council and the National Security Council—and that may dilute the office's ability to craft real actions instead of fighting incompatible bureaucratic goals. One of the chief weaknesses of the Bush administration's cybersecurity policy was its failure to ensure the cybersecurity boss was appointed at a high enough level to possess real clout—and, says Schneider, that could happen again.

"You have somebody who is no longer just talking to the president. In fact, whomever they appoint will be a servant of many masters," he says.

Schneider says finding a way to bring the foundations of accountability prevalent in law enforcement into cyberspace "is a big step because cyberspace has had this value system that is about anonymity. But cyberspace, when it was constituted, was not constituted with anything of consequence being controlled or anything of value being accessed that way."

"But there is a stupid way and a sensible way to make cyberspace accountable," says Schneider, "and if you use too broad a brush when you bring accountability to cyberspace, you will blow it badly. And the people who are worried about privacy have a reasonable basis to be worried, because it's easy to do it badly, which is why research needs to be done, and the process needs to be open and transparent."

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

DOI:10.1145/1536616.1536627

Tim Draper

Economic and Business Dimensions Entrepreneurship During a Slump

A contrarian's perspective on how entrepreneurial opportunities and innovation can thrive during an economic crisis.

This installment of Economic and Business Dimensions is written by Tim Draper, the founder of Draper Fisher Jurvetson (DFJ). Located in Silicon Valley, DFJ has funded over 500 firms since its founding in 1985, including Hotmail, Baidu, Skype, and many others. Draper is known for first successful use of "viral marketing" while funding Hotmail and travels the world giving speeches about entrepreneurship, from which this column is derived. Draper revels in being a contrarian about entrepreneurial opportunities during the current economic slump.

—Shane Greenstein, Viewpoints section board member

Now is the time for the true entrepreneur to shine. As economies worldwide collapse and firms retrench, people look for alternatives: cheaper alternatives, faster alternatives, easier alternatives, and smaller alternatives to what they are using now. To some observers this is a crisis. To an entrepreneur this is an opportunity.

Let me say it to all the entrepreneurs: If you are not taking advantage of this economic crisis, you are missing one of the greatest opportunities of your life. Where do you want to be in the near future?

Those of you who have lost your jobs have had the decision made for you. With no other alternative, and companies not hiring, you are forced to become an entrepreneur of one sort or another. At the very least you must reinvent yourself, perhaps go back to school, and emerge with a new mission in hand.

Those of you who still have your jobs, now is the time to rock the boat. When things are going smoothly, nobody wants to rock the boat. *Now* is your time. You should make sure you are breaking old systems, inventing fresh concepts, and seeking out new customers for your existing business. Use a crisis like this to change those things in the organization you always wanted to change.

A Big Crisis Should Not Go To Waste

Consider this list of companies: GE, IBM, Microsoft, Shell Oil, AT&T, Merck, Johnson & Johnson, Sun Microsystems, Skype, Kodak, Polaroid, HP, and Adobe. What do they all have in common? They all were started during an economic downturn, some during the Great Depression.

Why do recessions give birth to some of the greatest, longest lasting, and best-run companies of the world? There are a number of reasons. First, many managers think more creatively if circumstances push them into it. That is what happens as the economy approaches the nadir of the cycle (they also think creatively at the zenith, but that is a different story). In downturns managers will question old assumptions. The old ways don't work anymore. No concept is too crazy. They explore with new directions. Anything is possible. That leads to bigger bang for the buck. It leads to new experiments.

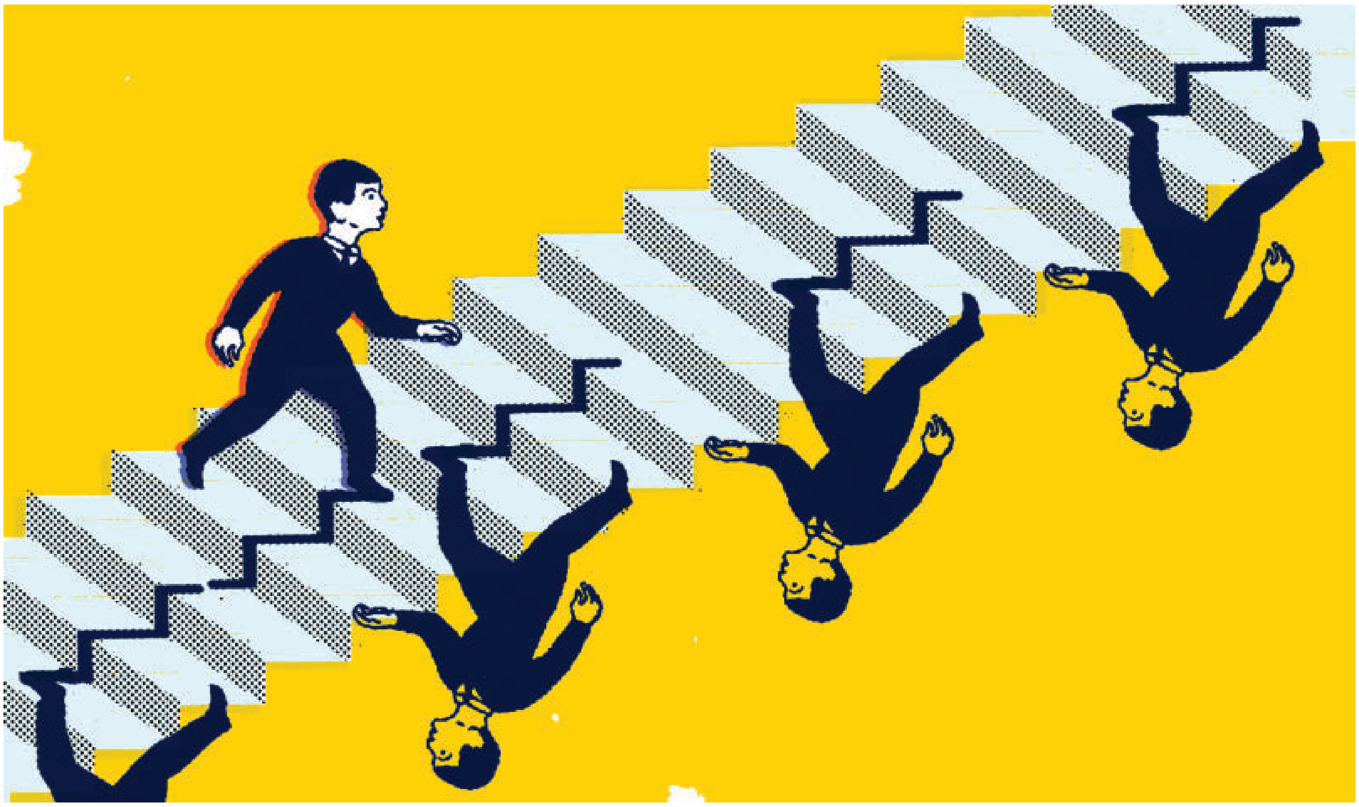


ILLUSTRATION BY YAREK WASZUL

Second, companies that start in downturns build long-lasting frugality into their cultures. During recessions people are willing to work for less. Managers are conscious of every penny they spend. And the frugality lasts once it is built in. For example, I began my career working at Hewlett-Packard. One day I took a few extra pencils from the box and was reminded by my peers of "profit sharing." Frugality was ingrained into every employee at Hewlett-Packard. The firm is one of the greatest companies of the world, and it has lasted for 60 years. Companies that last this long (or more) are always this frugal.

Third, managers in many established companies become shortsighted during downturns, and often their governing boards do too. These companies make shortsighted plans for product development, for research projects, and for company road maps. What do they do? They forget that the downturn will not last forever. They cut expenses today that would have helped their future. Then the future arrives and they find they have not prepared for it in advance. Worse, they are behind those who did prepare.

Fourth, during recessions entrepreneurs are not subject to what has become known as "venture fratricide." In the boom times, for example, my firm would fund companies and then discover that 20 competitors were funded by venture capital brethren. The companies would all fight over market share and spend tremendous amounts of money to win the market. This happened in the dot-com boom, where anything dot-com got funded. The new market was enormous, but enormous amounts of money were lost in the process.

To survive venture fratricide young firms specialize. Then markets became too narrowly defined to amount to anything big. For example, after someone started a flowers.com someone else started a roses.com, further segmenting an already small market. Established firms also specialize, but with their own versions of me-too divisions. This also happened with the dot-com boom. Every pure play online retailer eventually faced one or more branded retailer.

Why do recessions give birth to some of the greatest, longest lasting, and best-run companies in the world?

Today's recessionary markets do not have problems with fratricide or me-too divisions. Very few other groups are starting up with money, or going after anything new. Here is the message for a true entrepreneur: Since funding is not readily available, by necessity, an entrepreneur has to create a business model that works and is sustainable. If you start a business now, today it is an open green field.

Moore's Law Does Not Slow Down for Economic Downturns

Many commentators believe big ideas crop up once a decade or so. That is just wrong. Moore's Law marches on. The potential for innovation is growing exponentially and globally. The innovations and changes that occur in the next 10 to 15 years will change our lives as much as all the innovation that's happened in the last 50 to 100 years. If you are a technologist, the innovations you create today may well be the basis for enormous companies in the future. So work harder now. Now is the time. In a recession/depression, people need new directions. They need new directions politically, and they need new directions scientifically. Open your mind to new possibilities. Those possibilities can take the world in new directions that can enhance our lives economically, socially, and spiritually. Take a risk when no one else is trying, and you can set the path that others will follow for generations to come.

Big ideas crop up when scientists, engineers, entrepreneurs, and businesspeople go after big problems. A great entrepreneur with dedication, passion, and a grand vision has a good chance. Venture capital sees it over and over. Sometimes it is a by-product of taking a valuable first step. Sometimes the first step toward the grand vision yields enormous value. For example, yesterday's entrepreneurs sought the self-navigating cars and made GPS navigation affordable and accessible. Yesterday's entrepreneurs sought holo-decks and put virtual meetings within reach of the broadband world.

Sometimes it involves an unexpected bonanza. That is because a grand vision leads smart and dedicated entrepreneurs into new situations requiring novel solutions to challenging problems. That gives serendipity a chance to sparkle. What do electricity, penicillin, Velcro, Teflon, and microwave ovens have in common? They were all unexpected bonanzas. So was "viral marketing."

In 1996, I helped create an unexpected bonanza. An entrepreneurial team (Jack Smith and Sabeer Bhatia) came into DFJ with a proposal. They wanted to try something new: free Web-based email for the nascent and growing Internet. But it needed a way to attract new users. I recalled Tupperware's marketing from a case in business school, where one friend invites another. I suggested they put a line at the bottom of their email that made it easy for one friend to invite another. That was the start of viral marketing. Now viral marketing is standard practice for marketers everywhere.

Did we know it would work? We had a hunch, but it was a new situation and a novel problem. We could not be sure until a team of dedicated entrepreneurs gave it a try. Entrepreneurs who pursue a mission can find it leads to an unexpected bonanza. Jack Smith and Sabeer Bhatia were on a mission, which put me in a position to enhance their vision, which in turn drove a company to great heights.

Sometimes a third thing happens. Now and again, entrepreneurs do get what they wish for. There is a breakout winner that employs tens of thousands of people with very little investment dollars. It creates huge value and huge economies. If you are a technologist pursuing a vision, be prepared for innovations on top of your vision. The best technologies have taken many paths to get to market success.

In the future I expect to see a lot of valuable steps, unexpected bonanzas, and breakout winners. In fact, I am counting on it. Some recent investments of ours have brought on electric cars (Reva and Tesla), solar thermal generators (BrightSource), social networking for the enterprise (SocialText), iPhone GPS apps, new ways of communicating (Meebo), and nano solar panels (Konarka). Who knows what extraordinary businesses can be created in the future.

That is my advice for entrepreneurs in these times: If you take an entrepreneurial risk, make sure you go after something big. Extend your imagination. Think flying and self-navigating cars, holo-decks, brain enhancers, saltwater purifiers, fusion energy, and space travel.

Think Opportunity

I asked Warren Buffett what he thought we needed to do to get out of this economic quagmire. He said, "Throw a bunch of things up on the wall and see what sticks." He is absolutely right. Any experiment could pull us out this time too: A flying car, a cure for cancer, a better cellphone, brain-wave communications, entertainment on demand, fusion power, even a liquid stock market. Any of these could generate extraordinary demand and require a large work force.

Entrepreneurs and technologists will pull the economy out of the downturn, just as the growth of businesses around the Web pulled us out of an economic funk in the mid-1990s. Even if the businesses are only reasonably successful, or even if some of them fail, they move technology forward. They employ people. They get things happening. Their energy begets other ideas.

This is not the common mantra today. Bad news is good news to the press. Recently, reporters have seemingly gleefully given one account after another pointing toward an increasingly grim future. Companies are laying people off, markets are down worldwide, lending is frozen, real-estate values have plummeted, consumer and capital spending appears to have slowed, and investors are confused.

That misses the point. The future rests—and has always rested—squarely on the shoulders of entrepreneurs. Entrepreneurs create new jobs, they build value from nothing, they make our lives better, and they rebuild the economy.

It may sound contrarian, but it is not. I travel all over the world. In China they're thinking "opportunity" now. Even in a down world economy they're thinking "where's the opportunity?" I see it elsewhere too—in Ireland, Israel, Indonesia, and many other places.

Entrepreneurs with an intense dedication to what they are doing—one that will overcome all odds—seek to change the world. Their companies offer significant improvements over existing technology that could, if there is a large enough market, change the world. Eventually those entrepreneurs will help the economy boom.

During every financial downturn, from the Vienna stock exchange crash in 1873 through the Great Depression of the 1930s, the cold war of the 1950s, and the dot-com collapse that occurred earlier this decade, great entrepreneurs have embraced change. That attitude drives innovation and the economy to new successes. This time is no different. There's nothing that can stop the true entrepreneur from changing the world.

If you are that true entrepreneur, your time is now! Go for it! C

Tim Draper (Tim@dfj.com) is the founder and a managing director of Draper Fisher Jurvetson in Menlo Park, CA.

DOI:10.1145/1536616.1536628

Wanda Dann and Stephen Cooper

Education

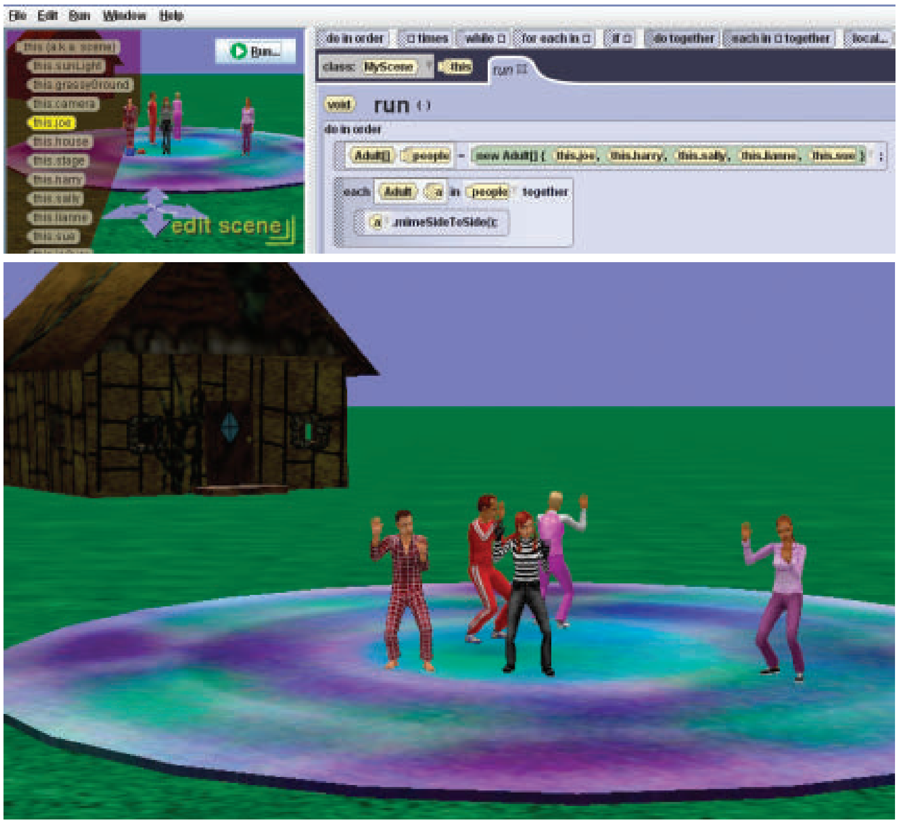
Alice 3: Concrete to Abstract

The innovative Alice 3 programming environment, currently in beta testing, teaches students to program with Alice and Java software.

COMPUTING EDUCATORS RECOGNIZE a critical need for innovative change to attract and maintain a stable and more diverse incoming enrollment. As noted by Peter Denning, "The loss of attraction to [computing] comes from our being unable to communicate the magic and beauty of the field."[5] The most recent Taulbee Survey[1] continues to indicate that women, Hispanics, and other traditionally underrepresented groups have not made enrollment gains over the last decade and diversity is at a record low. What innovative teaching and learning strategies might attract a more diverse student population and maintain students in undergraduate computing programs?

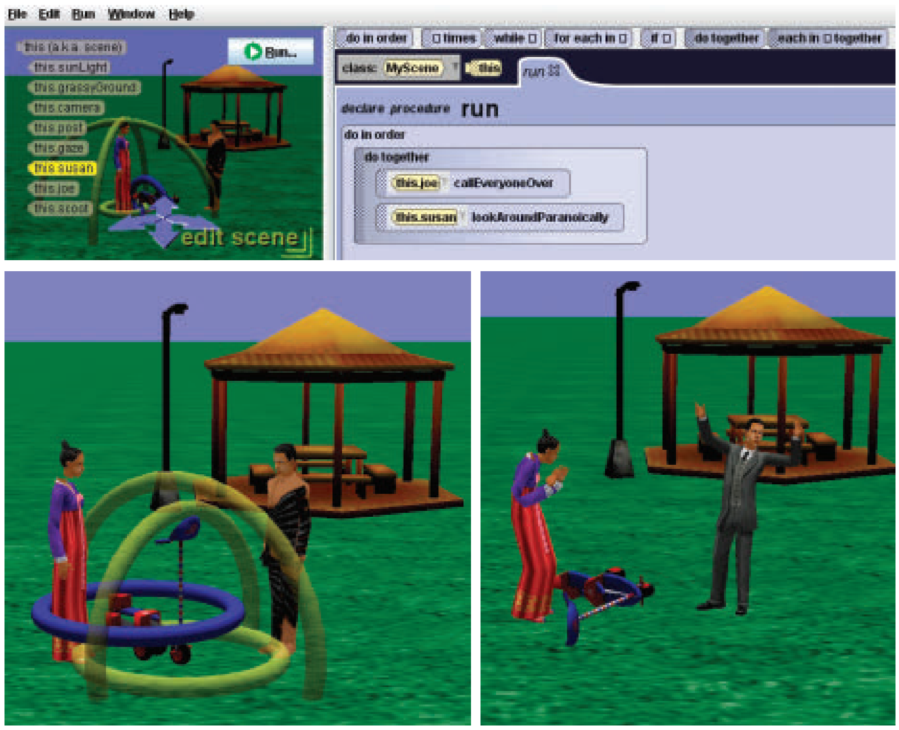
Educators might well take a lesson from colleagues in science and engineering, many of whom have struggled with diversity and decreasing enrollments in decades past. The American Association for the Advancement of Science (well known for the journal *Science*) has published *Science for All Americans*,[4] a highly praised book that describes science understandings all students should learn. Importantly, the authors devote an entire chapter to emphasize how science is taught is equally important with what is taught; that is, pedagogy matters. A particularly relevant quote is that students' learning progression is "usually from the concrete to the abstract. Young people can learn most readily about things that are tangible and directly accessible to their senses—visual, auditory, tactile, and kinesthetic. With experience, they grow in their ability to understand abstract concepts, manipulate symbols, reason logically, and generalize. These skills develop slowly, however, and the dependence of most people on concrete examples of new ideas persists throughout life. Concrete experiences are most effective in learning when they occur in the context of some relevant conceptual structure."

The assertions we've expressed here raise three interesting issues for computer science: the ubiquity of science concepts; the importance of context; and the movement from the concrete to the abstract. The first is related to aspects of science that all



```
void run ( )
do in order
  Adult[] people = new Adult[] { this.joe, this.harry, this.sally, this.lianne, this.sue } ;
  each Adult a in people together
    a .mimeSideToSide( );
```

Alice 3 array of Sims2 objects mime together.



```
declare procedure run
do in order
    do together
        this.joe callEveryoneOver
        this.susan lookAroundParanoically
```

Alice 3 code editor, scene editor, and runtime displays.

children must learn/know. Jeanette Wing, in her manifesto, Computational Thinking, argues analogously for the ubiquity of computation, as a skill that all must master.[6] She adds computational thinking as a fourth "analytical ability," to reading, writing, and arithmetic, which all young people should be exposed to and given the opportunity to master. Certainly, it is interesting to explore how Alice helps students gain capability in computational thinking abilities, and we describe our work with Alice in this column.

The second issue raised is the importance of context. We believe context helps to communicate the "magic and beauty" of our discipline. Exciting work is going on in this space. There is much excitement about the potential for robotics. From Lego Mindstorms to Scribblers to Myro and Chiara, the possibilities offer great potential. Guzdial and Ericson have discovered that working with media is fun for students who love to build their own (scaled down) versions of Photoshop and Audacity. Alice (in 3D) and Scratch (in 2D) use animation, storytelling, and game authorship as a context in which to entice students toward computing. LilyPad Arduino is doing much the same through e-textiles.

In many ways, the third issue is the most challenging for computing educators. Attracting a more diverse student population and maintaining students in undergraduate computing programs should involve a teaching and learning strategy that begins with the concrete in a context familiar to students and then gradually leads to an understanding of the abstract. Our discipline is all about abstraction. But it is not clear how to begin with the concrete and then move to the abstract, particularly in introductory computer programming courses that use languages inherently dependant on abstraction. Exciting innovations are gaining traction, however, through new pedagogic approaches such as Peer-Led Team-Learning, CS Unplugged, Supplemental Instruction, Project-Based Learning, and Problem-Based Learning.

Alice 2 brings a different form of magic and beauty into teaching and learning fundamental programming concepts.

These three issues are the driving force behind the creation of Alice 3, currently under development by the Alice team at Carnegie Mellon. Dennis Cosgrove, lead architect of Alice 2, is also the project scientist for developing the Alice 3 software.

A Brief History of Alice

Under the leadership of the late Randy Pausch, Alice was originally developed as a rapid prototyping tool for virtual reality animation, complete with head-mounted devices and glove sensors. Pausch used the original Alice in his Building Virtual Worlds course, where students from different disciplines (CS, design, and art) had to work together in small groups, under short deadlines, to build and demonstrate virtual worlds. The students thought they were learning about virtual reality, but they were also learning how to work together with other people in a way that others would respect them and keep working with them. Encouraging students to learn one thing while they believe they are learning something else is what Pausch called a "head fake." The course became so popular that it inspired a master's degree program in the Entertainment Technology Center at CMU.

In the late 1990s, we joined with the Alice team to pursue a dream of evolving Alice into an educational software tool that could be used to gently introduce students to computer programming. The goal was to create a new version of Alice that would introduce students to object-oriented programming concepts using concrete, visual objects in a 3D graphical environment. The concrete objects would be things that are familiar in their real world: houses, trees, animals, people, cellphones, but in a virtual world where a person can be scared by a spooky sound while riding a bicycle home after sports practice, and turn to see… imagination takes over…and students spend hours learning how to program.

The head fake is still there. Alice 2 provides a 3D programming environment with concrete objects that make it easy for students to create expressive animated movies, similar to film shorts created by professionals in animation studios such as Pixar and Disney. Students are engaged by the whimsical characters and get caught up in the excitement of creating animated movies or simple games. Students are learning how to create an animation, but also are learning about sequence, conditionals, Boolean expressions, repetition, and even concurrent execution.

Alice 2 brings a different form of magic and beauty into teaching and learning fundamental programming concepts as students build animated movies and interactive games. Alice's drag-and-drop editor prevents syntax errors that often frustrate beginning programmers. And, by engaging and stimulating students' creativity, Alice encourages students to invest more time on task. But does it work? Do students actually learn the concepts? Is it effective in retaining students?

The Alice approach consists of the current Alice 2 and *Learning to Program with Alice*[2] pedagogy and instructional materials, which were developed and disseminated with NSF support. To investigate the effectiveness of this approach, we carried out a pilot study, primarily targeting a pre-CS1 audience.[3] In this study, we saw that at-risk computing majors who were exposed to Alice were nearly twice as likely to continue on to CS2 as their control-group peers who had not been exposed to Alice (88% versus 47% retention). Beyond CS2, the at-risk students who had been exposed to Alice performed comparably to their non-at-risk peers. Unfortunately, the high attrition among at-risk students who had not been exposed to Alice resulted in too few control-group students remaining to meaningfully measure how they performed in later courses as compared to the students who had been exposed to Alice.

We have been tracking Alice adoption in colleges. To our knowledge, well over 10% of colleges and universities in the U.S. have adopted Alice and we are currently experiencing an explosion into high schools, as well as increasingly with schools abroad and in middle school. We distribute our monthly Alice e-newsletter to more than 2,000 educators (nearly half of whom are K–12 teachers, and more than 300 of whom have email addresses indicating they teach at institutions outside the U.S.).

The teacher can gradually lead students from the concrete context of animation to abstract data and structures in Java and a traditional context.

Moving From the Concrete to the Abstract

Alice 2 provides the concrete, but what about the abstract? No matter how much fun Alice 2 may be, computing students must eventually move to the abstract and into a production-level language such as Java, C++, Python, C#, or (your choice here). And, one message we have heard loud and clear from many computing educators at liberal arts and community colleges and at comprehensive universities is a demand for software, curriculum, and instructional materials that can be used to blend the Alice approach with Java in a regular CS1 course. One reason for this demand is the difficulty of adding a pre-CS1 course to curricula in liberal arts colleges where the number of courses that can be offered is limited by two factors: a small number of faculty available for teaching courses; and the need to balance the number of courses required for the major with a heavy load of required general education courses. Responding to this demand, Alice 3 targets CS1 or AP CS courses. (Alice 2 continues to be supported for pre-CS1 and CS0 courses.)

Alice 3 is currently undergoing extensive beta testing. Not only are we tracking bugs we are also obtaining feedback from teachers and students regarding three important features designed to support a concrete to abstract approach. The first is the teacher has a choice regarding how close to Java their students will begin. Two versions of the drag-and-drop interface are provided: one in a familiar, natural language style and with minimal syntax and one in actual Java code.

The second feature is more subtle. Animation programs constructed in Alice 2 use built-in motion methods, (such as move, turn, and roll). This means students sometimes need to focus attention on mechanics of turning an object's body joints to obtain a reasonable walk, skate, or peck (for a chicken) animation. Alice 3 provides two galleries with a richer set of animations: the Alice 2 characters and the Sims2 characters (contributed by Electronic Arts). Richer primitive animations enable students to design and program animations at a higher, more abstract level.

The third feature is a transition option that allows students to "open the hood" and type Java code, edit, and run it in the NetBeans text-based Java integrated development environment. This provides an ability to start in Alice's 3D animation environment and transition to programming in actual Java code. The teacher can gradually lead students from the concrete context of animation to abstract data and structures in Java and a traditional context.

The path toward helping students think "like a computer scientist... thinking at multiple levels of abstraction"[6] is an exciting and challenging one. We believe Alice 3 is one step along that path. **C**

References

1. Computing Research Association; http://www.cra.org/.
2. Dann, W.P., Cooper, S,. and Pausch, R. *Learning to Program with Alice.* Prentice Hall, 2006.
3. Moskal, B., Lurie, D., and Cooper, S. Evaluating the effectiveness of a new instructional approach. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (Norfolk, Virginia, March 3–7, 2004). SIGCSE '04. ACM, New York, NY, 75–79.
4. Rutherford, F.J. and Ahlgren, A. *Science for All Americans.* Second edition. Oxford University Press, 1991.
5. Violino, B. Time to reboot. *Commun. ACM 52*, 4 (Apr. 2009), 19.
6. Wing, J.M. Computational thinking. *Commun. ACM 49*, 3 (Mar. 2006), 33–35.

Wanda Dann (wpdann@andrew.cmu.edu) is the director of the Alice Project and an associate teaching professor in the computer science department at Carnegie Mellon University, Pittsburgh, PA.

Stephen Cooper (scooper@sju.edu) is an associate professor and the director of the Center for Visualization at Saint Joseph's University, Philadelphia, PA.

DOI:10.1145/1536616.1536630

Brian Snow and Clinton Brooks

Privacy and Security
An Ethics Code for U.S. Intelligence Officers

Debating and arguing the points of a proposed code of behavior to provide guidance in making choices can produce the most effective ethics training.

PHOTOGRAPH BY FLICKR USER LUCY SQUACCO

DOCTORS, ENGINEERS, LAWYERS, and many other occupations have formal ethics codes that help members of the professions make difficult choices. What about the U.S. intelligence community? There is a formal statement: "Principles of Ethical Conduct," found in Part 1 of Executive Order 12731 of October 17, 1990.[a] These 14 sentences bind all U.S. federal employees, but these sentences deal primarily with what is usually defined as fiscal matters—proper handling of government funds, nepotism, accepting lavish gifts, and other such ill-advised activities. These sentences can be found posted in the hallways of almost all U.S. federal buildings, including those of the U.S. intelligence community (IC).

As important as these sentences are, they fail to address what we are calling "mission ethics" for intelligence officers—what you can or cannot, should or should not, do in executing your IC agency's mission. Why should this be of concern?

The IC provides intelligence to U.S. policymakers and military commanders. The decisions they make based on such intelligence can have profound impacts on the U.S., the world, and individual citizens. Intelligence officers are keenly aware of this and strive to do their best, often operating in demanding, complex, and perhaps hazardous situations.

Many intelligence officers have had personal experiences dealing with exacting situations that had difficult ethical ramifications during their careers. Existing fiscal ethics were not germane, and any mission ethics that might have existed may differ by organization (even within an agency) and also may differ as a function of which particular group convened to make a decision. There can be significantly different ethical considerations between tactical and strategic missions.

Most IC employees are dedicated, honorable, and ethical public servants. But they function in secrecy and have power—a combination that is fraught with temptation. Some may have to suborn citizens of other nations or the agents of foreign governments. Some may perform operations that would be illegal if done against their citizens. Others may make decisions that can have significant unforeseen unintended consequences.

Several of us who have been career intelligence officers contend that we could have benefited greatly from a set of mission ethics studied and deliberated in training classes using case studies, away from our demanding work environment. Such study and discussion would enable contemplation, reflection, and internalization that could help an officer in the heat of action when others might be overwhelmed with expediency or emotion, or in other situations when perhaps faced with unethical opportunists (there are always a few such in any organization).

a See http://www.usoge.gov/laws_regs/exec_orders/eo12731.pdf

Ethics will not solve all problems or make decisions easy, but should help reduce the number of bad decisions that might be regretted later.

Could such a code be devised that would truly be useful? In October 2005, six IC employees and two corporate employees met to discuss this issue. By the end of the day we had decided devising a code was possible and that it would be well worth the effort.

The intelligence organizations of concern to us typically report to nation-states that have differing legal structures, cultures, and value systems. We restricted our attention to the U.S. intelligence community, which functions as an agent of the U.S government, a government designed to be responsive to its citizens and their values. Differing contexts would lead to differing ethics codes.

We agreed on several points:

▸ The code should be aspirational, not proscriptive. That is, it should outline behaviors to aspire to and not get into specific details of do's and don'ts.

▸ It should not be viewed as regulation or law, but as guidelines for making difficult decisions.

▸ It should be a set of short, easily understood sentences (for example, the Ten Commandments are not verbose).

▸ The statements must address at least the issues of lawfulness, transparency, accountability, truthfulness, examining consequences of planned actions, and protection of innocent individuals.

▸ It should be unclassified and explicitly made available to the public.

An unclassified code has many benefits. It keeps the code from being too detailed or focused on arcane intelligence matters such as sources and methods. It might also offer some protection against public outrage when classified actions become publicly known (and at times they will). If the citizenry accepted the published code language and the exposed action is in line with the public language, it could help citizens understand the rationale for the action and lessen adverse reactions or possibly offer an opportunity to further refine the language (and constraints on future actions) to be more in line with national values that may change over time.

We explored other ethics codes (including medical, legal, security, and military) as models, which helped us structure our code.

Also, being public may actually give the ethics code more "teeth" because employees will know that not just their bosses could be sitting in judgment on their actions. This fact might encourage junior employees to resist (however gently) bosses who might be ordering questionable actions, as well as reducing the number of times such bosses might be tempted to order such actions.

We knew the task of crafting actionable language of value would be difficult; we attempted a draft as an exercise to judge the difficulty. We explored other ethics codes (including medical, legal, security, and military) as models, which helped us structure our code. We started from fundamental principles. Our oath as federal employees to support the U.S. Constitution was a key factor and helped prioritize other allegiances we had, such as to bosses, organizations, and fellow citizens. For example, we decided the required oath-of-office we all take to support the U.S. Constitution also requires us to first serve our fellow citizens—since the Constitution's first words are: "We the People of the United States, in order to…".

We concluded that writing a code was possible, but not necessarily easy. We expanded the effort in December 2006 to a closed, invitation-only Internet discussion group of approximately 50 people (approximately one-third with substantial IC experience, one-third professional ethicists, and one-third other). As expected, the ensuing discussion was difficult work—

Draft: U.S. Intelligence Community "Mission Ethics" Projects

Preamble:

Intelligence work may present exceptional or unusual ethical dilemmas beyond those of ordinary life. Ethical thinking and review should be a part of our day to day efforts; it can protect our nation's and our agency's integrity, improve the chances of mission success, protect us from the consequences of bad choices, and preserve our alliances. Therefore, we adhere to the following standards of professional ethics and behavior:

▸ First, do no harm to U.S. citizens or their rights under the Constitution.

▸ We uphold the Constitution and the Rule of Law; we are constrained by both the spirit and the letter of the laws of the United States.

▸ We will comply with all international human rights agreements that our nation has ratified.

▸ We will insist on clarification of ambiguities that arise between directives or law and the principles of this code. We will protect those within our institutions who call reasonable attention to wrongdoing.

▸ Expediency is not an excuse for misconduct.

▸ We are accountable for our decisions and actions. We support timely, rigorous processes that fix accountability to the responsible person.

▸ Statements we make to our clients, colleagues, overseers and the U.S. public will be true, and structured not to unnecessarily mislead or conceal.

▸ We will resolve difficult ethical choices in favor of constitutional requirements, the truth, and our fellow citizens.

▸ We will address the potential consequences of our actions in advance, especially the consequences of failure, discovery, and unintended or collateral consequences of success.

▸ We will not impose unnecessary risk on innocents.

▸ Although we may work in secrecy, we will work so that when our efforts become known, our fellow citizens will be proud of us and of our efforts.

involving lengthy sessions on phrasing to clearly and properly capture the intended meaning succinctly, even for nuanced concepts.

By late 2008 we felt we had a reasonable draft (see the accompanying sidebar; note that in the draft sidebar content we have intentionally retained some minor structural flaws so that anyone actually considering adopting it would face at least minor edits, which could help trigger the deeper reflection we believe is needed by any organization considering adoption). Producing the draft drove home the point that the real value to all of us was the work in creating it, the forced reflection and reconsideration of beliefs, not the final text. We submit that the most effective ethics training will be achieved if officers engage in debating and arguing the points of the proposed code. Nevertheless, we think the text can be of great help to an employee trying to do the right thing in a specific intelligence circumstance; what would it demand of the employee? We did not try for precise wording that would cover all circumstances; instead, we strove to capture the intent and let it foster discussion, deliberation, and debate that would help people internalize the code.

The current draft is not perfect and certainly can be improved, but those of us who have wrestled with this think it is a good enough draft for the U.S. IC to seriously consider as a basis for further work that would lead to the adoption of such a code of mission ethics.

We invite the U.S. IC to undertake such an effort and hope that intelligence communities of other nations might consider it. Do not simply copy our words, but craft your own, suitable to you and created through concerted effort and discussion. Then develop training that will help you internalize them as a basis for action.

Share your code with the public: we think it will benefit you and gain support for, and acceptance of, your activities on behalf of your nation. C

Brian Snow is a retired U.S. National Security Agency executive-level technical director.

Clinton Brooks is a retired U.S. National Security Agency executive.

Please send comments to: MissionEthicsComments@comcast.net.

DOI:10.1145/1536616.1536631

Lance Fortnow

Viewpoint
Time for Computer Science to Grow Up

As the computer science field has evolved, so should the methods for disseminating computing research results.

Unlike every other academic field, computer science uses conferences rather than journals as the main publication venue. While this made sense for a young discipline, our field has matured and the conference model has fractured the discipline and skewered it toward short-term, deadline-driven research. Computer science should refocus the conference system on its primary purpose of bringing researchers together. We should use archive sites as the main method of quick paper dissemination and the journal system as the vehicle for advancing researchers' reputations.

In his May 2009 *Communications* Editor's Letter,[2] Moshe Vardi challenged the computer science community to rethink the major publication role conferences play in computer science. Here, I continue that discussion and strongly argue that the computer science field, now more than a half-century old, needs to adapt to a conference and journal model that has worked well for all other academic fields.

Why do we hold conferences?

- To rate publications and researchers.
- To disseminate new research results and ideas.
- To network, gossip, and recruit.
- To discuss controversial issues in the community.

The de facto main role of computer science conferences is the first item: rating papers and people. When we judge candidates for an academic position, we first check the quality and quantity of the conferences where their work has appeared. The current climate of conferences and program committees often leads to rather arbitrary decisions even though these choices can have a great impact particularly on researchers early in their academic careers.

But even worse, the focus on using conferences to rate papers has led to a great growth in the number of meetings. Most researchers don't have the time and/or money to travel to conferences where they do not have a paper. This greatly affects the other roles, as conferences no longer bring the community together and thus we are only disseminating, networking, and discussing with a tiny subset of the community. Other academic fields leave rating papers and researchers to academic journals, where one can have a more lengthy and detailed reviews of submissions. This leaves conferences to act as a broad forum and bring their communities together.

A Short History of CS Conferences

The growth of computers in the 1950s led nearly every major university to develop a strong computer science discipline over the next few decades. As a new field, computer science was free to experiment with novel approaches to publication not hampered by long traditions in more established scientific and engineering communities. Computer science came of age in the jet age where the time spent traveling to a conference no longer dominated the time spent at the conference itself. The quick development of this new field required rapid review and distribution of results. So the conference system quickly developed, serving the multiple purposes of the distribution of papers through proceedings, presentations, a stamp of approval, and bringing the community together.

With the possible exception of *Journal of the ACM*, journals in computer science have not received the prestige levels that conferences do. Only a fraction of conference papers eventually get published in polished and extended form in a journal. Some universities insist on journal papers for promotion and tenure but for the most part researchers feel they have little incentive for the effort of a journal submission.

As the field went through dramatic growth in the 1980s we started to see a shift in conferences. The major CS conferences could no longer accept most qualified research papers. Not only did these conferences raise the bar on acceptance but for the papers on the margin a preference for certain subareas emerged. Researchers from the top CS departments dominated the program committees and, not necessarily consciously, helped set the agenda with areas that helped their faculty, students, and graduates. Over the years these biases became part of the system and unofficially accepted behavior in the community.

As CS grew the major conferences became even more selective and could not accept all the quality papers in any specialized area. Many new specialized conferences and workshops arose and grew to capture these papers. We currently have approximately a dozen U.S.-based conferences in theoretical computer science alone. The large number of conferences has splintered our communities. Because of limitations of money and time, very few conferences draw many attendees beyond the authors of accepted papers. Conferences now serve the journal role of other fields, leaving nothing to serve the proper role of conferences.

Other disciplines have started to recognize the basic importance of computation and we have seen strong connections between CS and physics, biology, economics, mathematics, education, medicine and many other fields. Having different publication procedures discourages proper collaboration between researchers in CS and other fields.

The Current Situation

Most CS researchers would balk at paying significant page charges for a journal but think nothing of committing well over $1,000 for travel and registration fees for a conference if their paper were accepted (not to mention the time to attend the conference). What does that monetary commitment buy the author? A not particularly fair review process.

With the tremendous almost continual growth in computer science over the past half-century combined with the desire of each conference to remain small and "competitive," even with the increase in the number of conferences we simply have too many papers chasing too few conference slots. Each conference has a program committee that examines submissions and makes decisions on which papers will appear at a conference and which will not. The great papers almost always are accepted and the worst papers mostly get rejected. The problem occurs for the vast majority of solid papers landing in the middle. Conferences cannot accept all of these papers and still maintain their high-quality reputations.

Even if the best decisions are made, several good papers will not make the cut. A variety of factors make the process imperfect at best:

- Being on a program committee (PC) requires a large time commitment because of the number of papers involved. With the increase in conferences, many researchers, particularly senior scientists, cannot serve on many of these committees, leaving these important decisions mostly to those with less experience.
- As our research areas continue to become more specialized a few to none of the PC members can properly judge the importance of most results.
- These specialized areas have a small number of researchers, meaning the appropriate PC members know the authors involved and personal feelings can influence decision making.
- PC members tend to favor papers in their own areas.
- The most difficult decisions are made by consensus. This leads to an emphasis on safe papers (incremental and technical) versus those that explore new models and research directions outside the established core areas of the conference.
- No or limited discussions between authors and the PC means papers often get rejected for simple misunderstandings.

Various conferences have implemented a number of innovative and sometimes controversial ideas to try to make the process more fair (author information removed from papers, author responses to initial reviews, multilevel program committees, separate tracks for areas and quality, higher/lower acceptance ratios) but none can truly avoid most of the problems I've outlined here.

In the extreme many of the best scientific papers slip through the cracks. For example, nearly half of the Gödel Prize winners (given to the best CS theory papers after they've appeared in journals) were initially rejected or didn't appear at all in the top theoretical computer science conferences.

We end up living in a deadline-driven world, submitting a paper when we reach an appropriate conference deadline instead of when the research has been properly fleshed out. Many also just publish "least-publishable units," doing just enough to get accepted into a conference.

The Road Ahead

How do we move a field mired in a long tradition of conference publications to a more journal-based system? Computer science lacks a single strong central organization that can by itself break the inertia in our system.

The Computing Research Association, in its 1999 tenure policy memo,[1] specifically puts conference publications above journals: "The reason conference publication is preferred to journal publication, at least for experimentalists, is the shorter time to print (7 months vs. 1–2 years), the opportunity to describe the work before one's peers at a public presentation, and the more complete level of review (4–5 evaluations per paper compared to 2–3 for an archival journal). Publication in the prestige conferences is inferior to the prestige journals only in having significant page limitations and little time to polish the paper. In those dimensions that count most, conferences are superior."

A decade later the CRA should acknowledge that the growth in computer science and advances in technology changes the calculus of this argument. Quick dissemination via the Web makes time to print less relevant and two or three careful journal referee reports give a much more detailed level of review than four or five rushed evaluations of conference reviewers. The CRA needs to make a new statement that the current conference system no longer fully meets the needs of the computer science community and support the growth of a strong journal publication system. This will also encourage chairs and deans to base hiring and promotion more on journal publications as it should be.

Many of the strongest computer science conferences in the U.S. are sponsored by ACM and the IEEE Computer Society. These organizations need to allow special interest groups and technical committees to restructure or perhaps eliminate some conferences even if it hurts their publication portfolio and finances in the short term.

How do we move a field mired in a long tradition of conference publications to a more journal-based system?

But most importantly, leaders of major conferences must make the first move, holding their conferences less frequently and accepting every reasonable paper for presentation without proceedings. By de-emphasizing their publication role, conferences can once again play their most important role: Bringing the community together.

Conclusion

Our conference system forces researchers to focus too heavily on quick, technical, and safe papers instead of considering broader and newer ideas. Meanwhile, we have devoted much of our time and money to conferences where we can present our research that we can rarely attend conferences and workshops to work and socialize with our colleagues.

Computer science has grown to become a mature field where no major university can survive without a strong CS department. It is time for computer science to grow up and publish in a way that represents the major discipline it has become. C

References

1. Evaluating computer scientists and engineers for promotion and tenure; http://www.cra.org/reports/tenure_review.html.
2. Vardi, M. Conferences vs. journals in computing research. *Commun. ACM 52*, 5 (May 2009), 5.

Lance Fortnow (fortnow@eecs.northwestern.edu) is a professor of electrical engineering and computer science at Northwestern University, Evanston, IL.

I would like to thank Ming-Yang Kao, Richard Ladner, Rakesh Vohra, and the reviewers for their helpful comments and the many people with whom I've shared discussions on this topic. The views presented in this Viewpoint remain solely my own.

Calendar of Events

August 24–28
Joint 12th European Software Engineering Conference and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering, Amsterdam, Netherlands, Sponsored: SIGSOFT, Contact: J C van Vliet, Phone: 31-20-444-7768, Email: hans@cs.vu.nl

August 25–28
15th International Euro-par Conference, Netherlands, Contact: Henk J Sips, Phone: 31-15-2781670, Email: sips@cs.tudelft.np

August 27–28
Principles and Practice of Programming in Java, Alberto, Canada, Contact: Ben Stephenson, Email: ben@csd.uwo.ca

August 29–31
IEEE International Conference on Privacy, Vancouver, Canada, Contact: Justin Zhijun Zhan, Email: justinzh@andrew.cmu.edu

August 29–30
2nd International Workshop on Similarity Search and Applications, Prague, Czech Republic, Contact: Tomas Skopal, Email: skopal@ksi.mff.cuni.cz

August 29–31
IEEE International Conference on Social Computing, Vancouver, Canada, Contact: Justin Zhan, Email: justinzh@andrew.cmu.edu

September 1–5
British Computer Society Conference on Human-Nature Computer Interaction, Cambridge United Kingdom, Contact: Alan F. Blackwell, Email: afb21@cam.ac.uk

September 2–4
Workshop in Information Security Theory and Practices 2009: Smart Devices, Pervasive Systems, and Ubiquitous Networks, Brussels, Belgium, Contact: Oliver Markowitch, Email: Olivier.markowitch@ulb.ac.be

DOI:10.1145/1536616.1536632

Article development led by acmqueue
queue.acm.org

Scale up your datasets enough and your apps come undone. What are the typical problems and where do the bottlenecks surface?

BY ADAM JACOBS

The Pathologies of Big Data

WHAT IS "BIG DATA" anyway? Gigabytes? Terabytes? Petabytes? A brief personal memory may provide some perspective. In the late 1980s at Columbia University, I had the chance to play around with what at the time was a truly enormous disk: the IBM 3850 MSS (Mass Storage System). The MSS was actually a fully automatic robotic tape library and associated staging disks to make random access, if not exactly instantaneous, at least fully transparent. In Columbia's configuration, it stored a total of around 100GB. It was already on its way out by the time I got my hands on it, but in its heyday, the early- to mid-1980s, it had been used to support access by social scientists to what was unquestionably "big data" at the time: the entire 1980 U.S. Census database.[2]

Presumably, there was no other practical way to provide the researchers with ready access to a dataset that large—at close to $40K per GB,[3] a 100GB disk farm would have been far too expensive, and requiring the operators to manually mount and dismount thousands of 40MB tapes would have slowed progress to a crawl, or at the very least severely limited the kinds of questions that could be asked about the census data.

A database on the order of 100GB would not be considered trivially small even today, although hard drives capable of storing 10 times as much can be had for less than $100 at any computer store. The U.S. Census database included many different datasets of varying sizes, but let's simplify a bit: 100GB is enough to store at least the basic demographic information—age, sex, income, ethnicity, language, religion, housing status, and location, packed in a 128-bit record—for every living human being on the planet. This would create a table of 6.75 billion rows and maybe 10 columns. Should that still be considered "big data?" It depends, of course, on what you're trying to do with it. Certainly, you could *store* it on $10 worth of disk. More importantly, any competent programmer could in a few hours write a simple, unoptimized application on a $500 desktop PC with minimal CPU and RAM that could crunch through that dataset and return answers to simple aggregation queries such as "what is the median age by sex for each country?" with perfectly reasonable performance.

To demonstrate this, I tried it, with fake data of course—namely, a file consisting of 6.75 billion 16-byte records containing uniformly distributed random data (see Figure 1). Since a 7-bit age field allows a maximum of 128 possible values, one bit for sex allows only two (we'll assume there were no NULLs), and eight bits for country allows up to 256 (the UN has 192 member states), we can calculate the

Details of Jason Salavon's 2008 data visualization American Varietal (U.S. Population, by County, 1790–2000), commissioned as part of a site-specific installation for the U.S. Census Bureau. http://salavon.com/

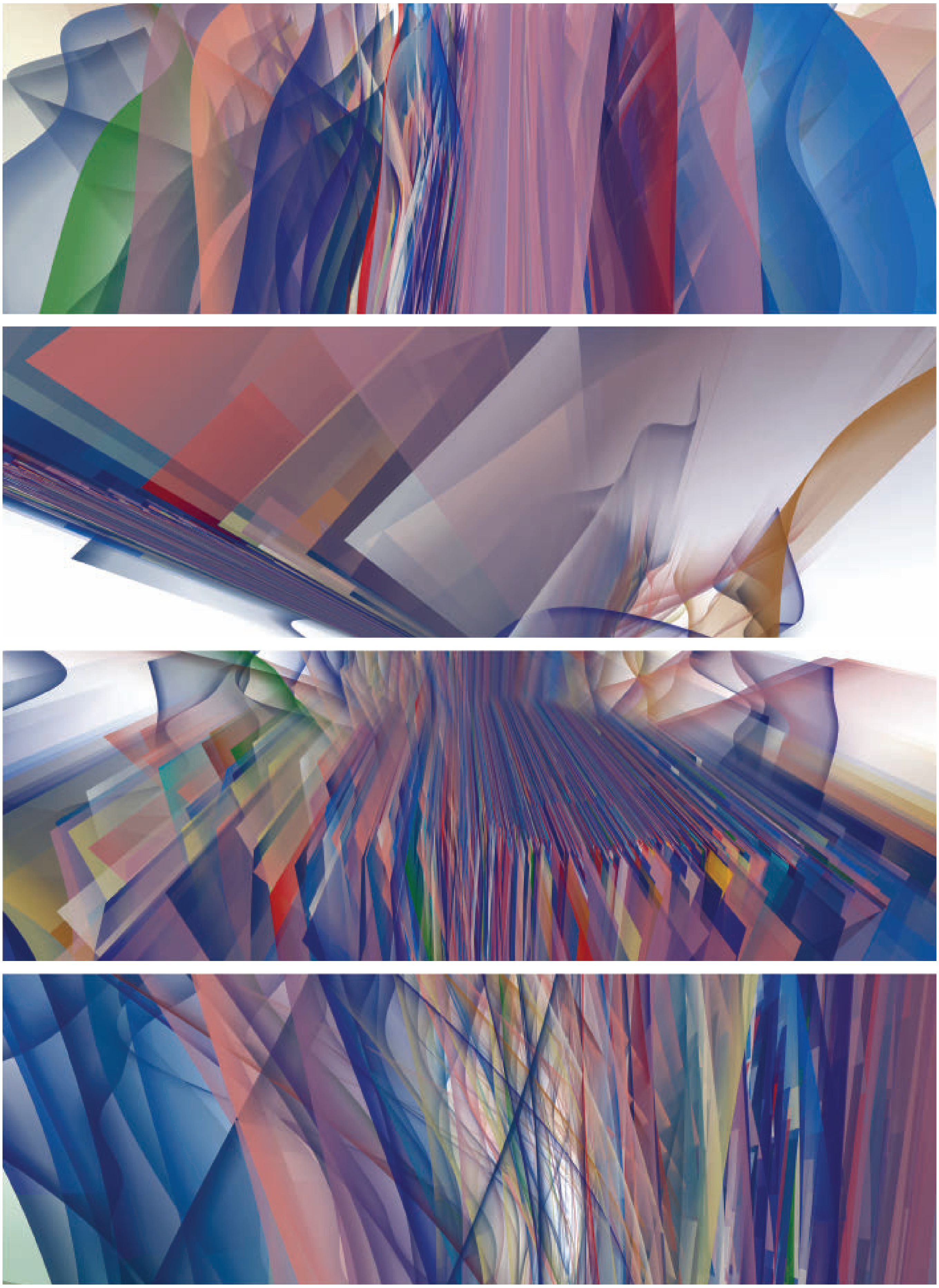
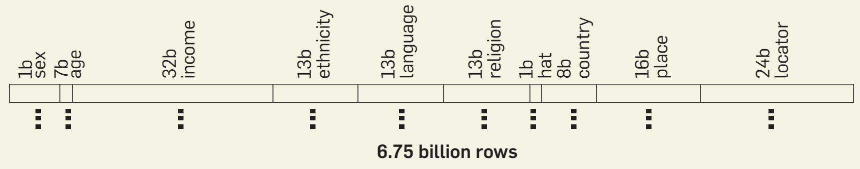


Figure 1. Calculating the median age by sex and country over the entire world population in a matter of minutes.

Record layout:



To find median age by sex and country,

```
int age, sex, country;
int cnt[2][256][128];
int tot,acc;
byte r[16];
fill cnt with 0;
do
   read 16 bytes into r;
   age = r[0] & 01111111b;
   sex = r[1] & 10000000b;
   ctry = r[11] & 11111111b;
   cnt[sex][ctry][age] += 1;
until end of file;
```

then

```
for sex = 0 to 1 do
 for ctry = 0 to 255 do
   output ctry, sex;
   tot = sum9cnt[sex][ctry][age];
   acc = 0;
   for age = 0 to 127 do
     acc += cnt[sex][ctry][age];
     if(acc >= tot/2)
       output age;
       go to next ctry;
     end if;
   next age;
 next ctry;
next sex;
```

median age by using a counting strategy: simply create 65,536 buckets—one for each combination of age, sex, and country—and count how many records fall into each. We find the median age by determining, for each sex and country group, the cumulative count over the 128 age buckets: the median is the bucket where the count reaches half of the total. In my tests, this algorithm was limited primarily by the speed at which data could be fetched from disk: a little over 15 minutes for one pass through the data at a typical 90MB/s sustained read speed,[9] shamefully underutilizing the CPU the whole time.

In fact, our table of "all the people in the world" will fit in the *memory* of a single, $15K Dell server with 128GB RAM. Running off in-memory data, my simple median-age-by-sex-and-country program completed in less than a minute. By such measures, I would hesitate to call this "big data," particularly in a world where a single research site, the LHC (Large Hadron Collider) at CERN (European Organization for Nuclear Research), is expected to produce 150,000 times as much raw data each year.[10]

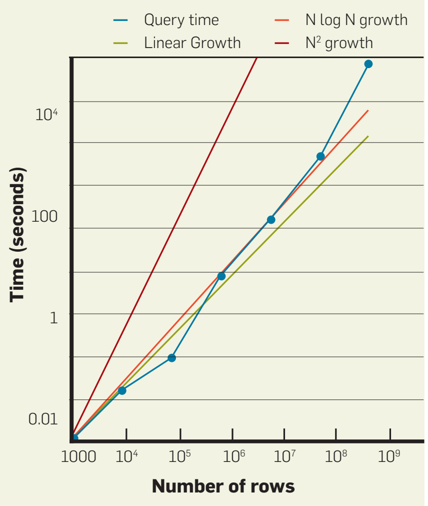
For many commonly used applications, however, our hypothetical 6.75-billion-row dataset would in fact pose a significant challenge. I tried loading my fake 100GB world census into a commonly used enterprise-grade database system (PostgreSQL[6]) running on relatively hefty hardware (an eight-core Mac Pro workstation with 20GB RAM and two terabytes of RAID 0 disk), but had to abort the bulk load process after six hours as the database storage had already reached many times the size of the original binary dataset, and the workstation's disk was nearly full. (Part of this, of course, was a result of the "unpacking" of the data. The original file stored fields bit-packed rather than as distinct integer fields, but subsequent tests revealed that the database was using three to four times as much storage as would be necessary to store each field as a 32-bit integer. This sort of data "inflation" is typical of a traditional RDBMS and shouldn't necessarily be seen as a problem, especially to the extent that it is part of a strategy to improve performance. After all, disk space is relatively cheap.)

I was successfully able to load subsets consisting of up to one billion rows of just three columns: country (8-bits, 256 possible values), age (7-bits, 128 possible values), and sex (one bit, two values). This was only 2% of the raw data, although it ended up consuming more than 40GB in the DBMS. I then tested the following query, essentially the same computation as the left side of Figure 1:

```
SELECT country,age,sex,count(*)
FROM people GROUP BY
country,age,sex;
```

This query ran in a matter of seconds on small subsets of the data, but execution time increased rapidly as the number of rows grew past 1 million (see Figure 2). Applied to the entire billion rows, the query took more than 24 hours, suggesting that PostgreSQL was not scaling gracefully to this big dataset, presumably because of a poor choice of algorithm for the given data and query. Invoking the DBMS's built-in EXPLAIN facility revealed the problem: while the query planner chose a reasonable hash table-based aggregation strategy for small tables, on larger tables it switched to sorting by grouping columns—a viable, if sub-optimal strategy given a few million rows, but a very poor one when facing a billion. PostgreSQL tracks statistics such as the minimum and maximum value of each column in a table (and I verified that it had correctly identified the ranges of all three columns), so it could have chosen a hash-table strategy with confidence. It's worth noting, however, that even if the table's statistics had not been known, on a billion rows it would take far less time to do an initial scan and determine

Figure 2. PostgreSQL performance on the query SELECT country,age,sex,count(*) FROM people GROUP BY country,age,sex.



* Curves of linear, linearithmic, and quadratic growth are shown for comparison.

the distributions than to embark on a full-table sort.

PostgreSQL's difficulty here was in analyzing the stored data, not in storing it. The database didn't blink at loading or maintaining a database of a billion records; presumably there would have been no difficulty storing the entire 6.75-billion-row, 10-column table had I had sufficient free disk space.

Here's the big truth about big data in traditional databases: it's easier to get the data in than out. Most DBMSs are designed for efficient transaction processing: adding, updating, searching for, and retrieving small amounts of information in a large database. Data is typically *acquired* in a transactional fashion: imagine a user logging into a retail Web site (account data is retrieved; session information is added to a log), searching for products (product data is searched for and retrieved; more session information is acquired), and making a purchase (details are inserted in an order database; user information is updated). A fair amount of data has been added effortlessly to a database that—if it's a large site that has been in operation for a while—probably already constitutes "big data."

There is no pathology here; this story is repeated in countless ways, every second of the day, all over the world. The trouble comes when we want to take that accumulated data, collected over months or years, and learn something from it—and naturally we want the answer in seconds or minutes! The pathologies of big data are primarily those of analysis. This may be a slightly controversial assertion, but I would argue that transaction processing and data storage are largely solved problems. Short of LHC-scale science, few enterprises generate data at such a rate that acquiring and storing it pose major challenges today.

In business applications, at least, data warehousing is ordinarily regarded as the solution to the database problem (data goes in but doesn't come out). A data warehouse has been classically defined as "a copy of transaction data specifically structured for query and analysis,"[4] and the general approach is commonly understood to be bulk extraction of the data from an operational database, followed by reconstitution in a different database in a form that is more suitable for analytical queries (the so-called "extract, transform, load," or sometimes "extract, load, transform" process). Merely saying, "We will build a data warehouse" is not sufficient when faced with a truly huge accumulation of data.

How must data be structured for query and analysis, and how must analytical databases and tools be designed to handle it efficiently? Big data changes the answers to these questions, as traditional techniques such as RDBMS-based dimensional modeling and cube-based OLAP (online analytical processing) turn out to be either too slow or too limited to support asking the really interesting questions about warehoused data. To understand how to avoid the pathologies of big data, whether in the context of a data warehouse or in the physical or social sciences, we need to consider what really makes it "big."

To understand how to avoid the pathologies of big data, whether in the context of a data warehouse or in the physical or social sciences, we need to consider what really makes it "big."

Dealing with Big Data

Data means "things given" in Latin—although we tend to use it as a mass noun in English, as if it denotes a substance—and ultimately, almost all useful data is given to us either by nature, as a reward for careful observation of physical processes, or by other people, usually inadvertently (consider logs of Web hits or retail transactions, both common sources of big data). As a result, in the real world, data is not just a big set of random numbers; it tends to exhibit predictable characteristics. For one thing, as a rule, the largest *cardinalities* of most datasets—specifically, the number of distinct entities about which observations are made—are small compared with the total number of observations.

This is hardly surprising. Human beings are making the observations, or being observed as the case may be, and there are no more than 6.75 billion of them at the moment, which sets a rather practical upper bound. The objects about which we collect data, if they are of the human world—Web pages, stores, products, accounts, securities, countries, cities, houses, phones, IP addresses—tend to be fewer in number than the total world population. Even in scientific datasets, a practical limit on cardinalities is often set by such factors as the number of available sensors (a state-of-the-art neurophysiology dataset, for example, might reflect 512 channels of recording[5]) or simply the number of distinct entities that humans have been able to detect and identify (the largest astronomical catalogs, for example, include several hundred million objects[8]).

What makes most big data *big* is repeated observations over time and/or space. The Web log records millions of visits a day to a handful of pages; the cellphone database stores time and location every 15 seconds for each of a few million phones; the retailer has thousands of stores, tens of thousands of products, and millions of customers but logs billions and billions of individual transactions in a year. Scientific measurements are often made at a high time resolution (thousands of samples a second in neurophysiology, far more in particle physics) and really start to get huge when they involve two or three dimensions of space as well; fMRI neuroimaging studies can generate hundreds or even thousands of gigabytes in a single experiment. Imaging in general is the source of some of the biggest big data out there, but the problems of large image data are a topic for an article by themselves; I won't consider them further here.

The fact that most large datasets have inherent temporal or spatial dimensions, or both, is crucial to understanding one important way that big data can cause performance problems, especially when databases are involved. It would seem intuitively obvious that data with a time dimension, for example, should in most cases be stored and processed with at least a partial temporal ordering to preserve locality of reference as much as possible when data is consumed in time order. After all, most nontrivial analyses will involve at the very least an aggregation of observations over one or more contiguous time intervals. One is more likely, for example, to be looking at the purchases of a randomly selected set of customers over a particular time period than of a "contiguous range" of customers (however defined) at a randomly selected set of times.

Here's the big truth about big data in traditional databases: It's easier to get the data in than out.

The point is even clearer when we consider the demands of time-series analysis and forecasting, which aggregate data in an order-dependent manner (for example, cumulative and moving-window functions, lead and lag operators, among others). Such analyses are necessary for answering most of the truly interesting questions about temporal data, broadly: "What happened?" "Why did it happen?" "What's going to happen next?"

The prevailing database model today, however, is the relational database, and this model explicitly ignores the ordering of rows in tables.[1] Database implementations that follow this model, eschewing the idea of an inherent order on tables, will inevitably end up retrieving data in a nonsequential fashion once it grows large enough that it no longer fits in memory. As the total amount of data stored in the database grows, the problem only becomes more significant. *To achieve acceptable performance for highly order-dependent queries on truly large data, one must be willing to consider abandoning the purely relational database model* for one that recognizes the concept of inherent ordering of data down to the implementation level. Fortunately, this point is slowly starting to be recognized in the analytical database sphere.

Not only in databases, but also in application programming in general, big data greatly magnifies the performance impact of suboptimal access patterns. As dataset sizes grow, it becomes increasingly important to choose algorithms that exploit the efficiency of sequential access as much as possible at all stages of processing. Aside from the obvious point that a 10:1 increase in processing time (which could easily result from a high proportion of nonsequential accesses) is far more painful when the units are hours than when they are seconds, increasing data sizes mean that data access becomes less and less efficient. The penalty for inefficient access patterns increases disproportionately as the limits of successive stages of hardware are exhausted: from processor cache to memory, memory to local disk, and—rarely nowadays!—disk to off-line storage.

On typical server hardware today, completely random memory access on a range much larger than cache size can be an order of magnitude or more slower than purely sequential access, but completely random disk access can be five orders of magnitude slower than sequential access (see Figure 3). Even state-of-the-art solid-state (flash) disks, although they have much lower seek latency than magnetic disks, can differ in speed by roughly four orders of magnitude between random and sequential access patterns. The results for the test shown in Figure 3 are the number of four-byte integer values read per second from a 1-billion-long (4GB) array on disk or in memory; random disk reads are for 10,000 indices chosen at random between one and one billion.

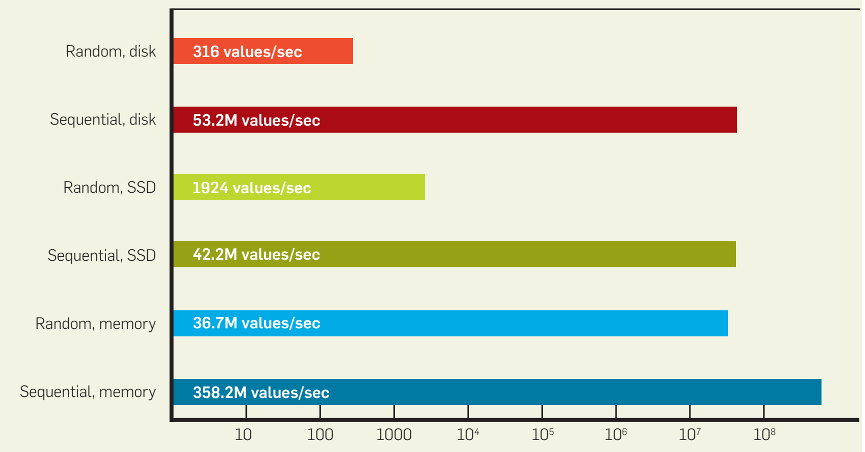
A further point that's widely underappreciated: in modern systems, as demonstrated in the figure, random access to memory is typically slower than sequential access to disk. Note that random reads from disk are more than 150,000 times slower than sequential access; SSD improves on this ratio by less than one order of magnitude. In a very real sense, *all* of the modern forms of storage improve only in degree, not in their essential nature, upon that most venerable and sequential of storage media: the tape.

The huge cost of random access has major implications for analysis of large datasets (whereas it is typically mitigated by various kinds of caching when data sizes are small). Consider, for example, joining large tables that are not both stored and sorted by the join key—say, a series of Web transactions and a list of user/account information. The transaction table has been stored in time order, both because that is the way the data was gathered and because the analysis of interest (tracking navigation paths, say) is inherently temporal. The user table, of course, has no temporal dimension.

As records from the transaction table are consumed in temporal order, accesses to the joined user table will be effectively random—at great cost if the table is large and stored on disk. If sufficient memory is available to hold the user table, performance will be improved by keeping it there. Because random access in RAM is itself expensive, and RAM is a scarce resource that may simply not be available for caching large tables, the best solution when constructing a large database for analytical purposes (for example, in a data warehouse) may, surprisingly, be to build a fully denormalized table—that is, a table including each transaction along with all user information that is relevant to the analysis (as shown in Figure 4).

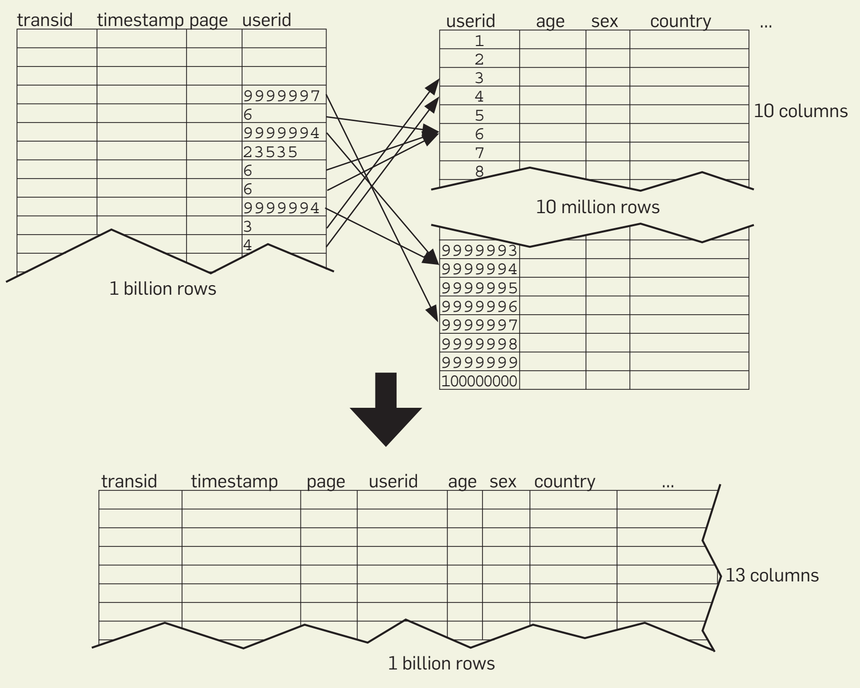
Denormalizing a 10-million-row, 10-column user information table onto a 1-billion-row, four-column transaction table adds substantially to the size of data that must be stored (the denormalized table is more than three times the size of the original tables combined). If data analysis is carried out in timestamp order but requires information from both tables,

Figure 3. Comparing random and sequential access in disk and memory.



* Disk tests were carried out on a freshly booted machine (a Windows 2003 server with 64GB RAM and eight 15,000RPM SAS disks in RAID5 configuration) to eliminate the effect of operating-system disk caching. SSD test used a latest generation Intel high-performance SATA SSD.

Figure 4. Denormalizing a user information table.



then eliminating random look-ups in the user table can improve performance greatly. Although this inevitably requires much more storage and, more importantly, more data to be read from disk in the course of the analysis, the advantage gained by doing all data access in sequential order is often enormous.

Hard Limits

Another major challenge for data analysis is exemplified by applications with hard limits on the size of data they can handle. Here, one is dealing mostly with the end-user analytical applications that constitute the last stage in analysis. Occasionally the limits are relatively arbitrary; consider the 256-column, 65,536-row bound on worksheet size in all versions of Microsoft Excel prior to the most recent one. Such a limit might have seemed reasonable in the days when main RAM was measured in megabytes, but it was clearly obsolete by 2007 when Microsoft updated Excel to accommodate up to 16,384 columns and one million rows. Enough for anyone? Excel is not targeted at users crunching truly huge datasets, but the fact remains that anyone working with a one million-row dataset (a list of customers along with their total purchases for a large chain store, perhaps) is likely to face a two million-row dataset sooner or later, and Excel has placed itself out of the running for the job.

In designing applications to handle ever-increasing amounts of data, developers would do well to remember that hardware specs are improving too, and keep in mind the so-called ZOI (zero-one-infinity) rule, which states that a program should "allow none of foo, one of foo, or any number of foo."[11] That is, limits should not be arbitrary; ideally, one should be able to do as much with software as the hardware platform allows.

Of course, hardware—chiefly memory and CPU limitations—is often a major factor in software limits on dataset size. Many applications are designed to read entire datasets into memory and work with them there; a good example of this is the popular statistical computing environment R.[7] Memory-bound applications naturally exhibit higher performance than disk-bound ones (at least insofar as the data-crunching they carry out advances beyond single-pass, purely sequential processing), but requiring all data to fit in memory means that if you have a dataset larger than your installed RAM, you're out of luck. On most hardware platforms, there's a much harder limit on memory expansion than disk expansion: the motherboard has only so many slots to fill.

Data replicated to improve the efficiency of different kinds of analyses can also provide redundancy against the inevitable node failure.

The problem often goes further than this, however. Like most other aspects of computer hardware, maximum memory capacities increase with time; 32GB is no longer a rare configuration for a desktop workstation, and servers are frequently configured with far more than that. There is no guarantee, however, that a memory-bound application will be able to use all installed RAM. Even under modern 64-bit operating systems, many applications today (for example, R under Windows) have only 32-bit executables and are limited to 4GB address spaces—this often translates into a 2- or 3GB working set limitation.

Finally, even where a 64-bit binary is available—removing the absolute address space limitation—all too often relics from the age of 32-bit code still pervade software, particularly in the use of 32-bit integers to index array elements. Thus, for example, 64-bit versions of R (available for Linux and Mac) use signed 32-bit integers to represent lengths, limiting data frames to at most $2^{31}-1$, or about two billion rows. Even on a 64-bit system with sufficient RAM to hold the data, therefore, a 6.75-billion-row dataset such as the earlier world census example ends up being too big for R to handle.

Distributed Computing as a Strategy for Big Data

Any given computer has a series of absolute and practical limits: memory size, disk size, processor speed, and so on. When one of these limits is exhausted, we lean on the next one, but at a performance cost: an in-memory database is faster than an on-disk one, but a PC with 2GB RAM cannot store a 100GB dataset entirely in memory; a server with 128GB RAM can, but the data may well grow to 200GB before the next generation of servers with twice the memory slots comes out.

The beauty of today's mainstream computer hardware, though, is that it's cheap and almost infinitely replicable. Today it is much more cost-effective to purchase eight off-the-shelf, "commodity" servers with eight processing cores and 128GB of RAM each than it is to acquire a single system with 64 processors and a terabyte of RAM. Although the absolute numbers will change over time, barring a radical change in computer architectures, the general principle is likely to remain true for the foreseeable future. Thus, it's not surprising that distributed computing is the most successful strategy known for analyzing very large datasets.

Distributing analysis over multiple computers has significant performance costs: even with gigabit and 10-gigabit Ethernet, both bandwidth (sequential access speed) and latency (thus, random access speed) are several orders of magnitude worse than RAM. At the same time, however, the highest-speed local network technologies have now surpassed most locally attached disk systems with respect to bandwidth, and network latency is naturally much lower than disk latency.

As a result, the performance cost of storing and retrieving data on other nodes in a network is comparable to (and in the case of random access, potentially far less than) the cost of using disk. Once a large dataset has been distributed to multiple nodes in this way, however, a huge advantage can be obtained by distributing the *processing* as well—so long as the analysis is amenable to parallel processing.

Much has been and can be said about this topic, but in the context of a distributed large dataset, the criteria are essentially related to those discussed earlier: just as maintaining locality of reference via sequential access is crucial to processes that rely on disk I/O (because disk seeks are expensive), so too, in distributed analysis, processing must include a significant component that is local in the data—that is, does not require simultaneous processing of many disparate parts of the dataset (because communication between the different processing domains is expensive). Fortunately, most real-world data analysis does include such a component. Operations such as searching, counting, partial aggregation, record-wise combinations of multiple fields, and many time-series analyses (if the data is stored in the correct order) can be carried out on each computing node independently.

Figure 5. Two ways to distribute 10 years of sensor data for 1,000 sites over 10 machines.

Node 1

timestamp	sensor	reading
19990101000000	1	
19990101000015	1	
19990101000030	1	
⋮	⋮	⋮
20081231235930	1	
20081231235945	1	
19990101000000	2	
19990101000015	2	
19990101000030	2	
⋮	⋮	⋮
20081231235930	2	
20081231235945	2	
19990101000000	3	
⋮	⋮	⋮
20081231235945	100	

Node 2

timestamp	sensor	reading
19990101000000	101	
19990101000015	101	
19990101000030	101	
⋮	⋮	⋮
20081231235930	101	
20081231235945	101	
19990101000000	102	
19990101000015	102	
19990101000030	102	
⋮	⋮	⋮
20081231235930	102	
20081231235945	102	
19990101000000	103	
⋮	⋮	⋮
20081231235945	200	

Node 10

timestamp	sensor	reading
19990101000000	901	
19990101000015	901	
19990101000030	901	
⋮	⋮	⋮
20081231235930	901	
20081231235945	901	
19990101000000	902	
19990101000015	902	
19990101000030	902	
⋮	⋮	⋮
20081231235930	902	
20081231235945	902	
19990101000000	903	
⋮	⋮	⋮
20081231235945	1000	

Node 1

timestamp	sensor	reading
19990101000000	1	
19990101000000	2	
19990101000000	3	
⋮	⋮	⋮
19990101000000	1000	
19990101000015	1	
19990101000015	2	
19990101000015	3	
19990101000015	4	
⋮	⋮	⋮
19990101000015	1000	
19990101000030	1	
19990101000030	2	
⋮	⋮	⋮
19991231235945	100	

Node 2

timestamp	sensor	reading
20000101000000	1	
20000101000000	2	
20000101000000	3	
⋮	⋮	⋮
20000101000000	1000	
20000101000015	1	
20000101000015	2	
20000101000015	3	
20000101000015	4	
⋮	⋮	⋮
20000101000015	1000	
20000101000030	1	
20000101000030	2	
⋮	⋮	⋮
20001231235945	1000	

Node 10

timestamp	sensor	reading
20080101000000	1	
20080101000000	2	
20080101000000	3	
⋮	⋮	⋮
20080101000000	1000	
20080101000015	1	
20080101000015	2	
20080101000015	3	
20080101000015	4	
⋮	⋮	⋮
20080101000015	1000	
20080101000030	1	
20080101000030	2	
⋮	⋮	⋮
20081231235945	1000	

Furthermore, where communication between nodes is required, it often occurs after data has been extensively aggregated; consider, for example, taking an average of billions of rows of data stored on multiple nodes. Each node is required to communicate only two values—a sum and a count—to the node that produces the final result. Not every aggregation can be computed so simply, as a global aggregation of local sub-aggregations (consider the task of finding a global median, for example, instead of a mean), but many of the important ones can, and there are distributed algorithms for other, more complicated tasks that minimize communication between nodes.

Naturally, distributed analysis of big data comes with its own set of "gotchas." One of the major problems is nonuniform distribution of work across nodes. Ideally, each node will have the same amount of independent computation to do before results are consolidated across nodes. If this is not the case, then the node with the most work will dictate how long we must wait for the results, and this will obviously be longer than we would have waited had work been distributed uniformly; in the worst case, all the work may be concentrated in a single node and we will get no benefit at all from parallelism.

Whether this is a problem or not will tend to be determined by how the data is distributed across nodes; unfortunately, in many cases this can come into direct conflict with the imperative to distribute data in such a way that processing at each node is local. Consider, for example, a dataset that consists of 10 years of observations collected at 15-second intervals from 1,000 sensor sites. There are more than 20 million observations for each site; and, because the typical analysis would involve time-series calculations—say, looking for unusual values relative to a moving average and standard deviation—we decide to store the data ordered by time for each sensor site (shown in Figure 5), distributed over 10 computing nodes so that each one gets all the observations for 100 sites (a total of two billion observations per node). Unfortunately, this means that whenever we are interested in the results of only one or a few sensors, most of our computing nodes will be totally idle. Whether the rows are clustered by sensor or by time stamp makes a big difference in the degree of parallelism with which different queries will execute.

We could, of course, store the data ordered by time, one year per node, so that each sensor site is represented in each node (we would need some communication between successive nodes at the beginning of the computation to "prime" the time-series calculations). This approach also runs into the difficulty if we suddenly need an intensive analysis of the past year's worth of data. Storing the data *both* ways would provide optimal efficiency for both kinds of analysis—but the larger the dataset, the more likely it is that two copies would be simply too much data for the available hardware resources.

Another important issue with distributed systems is reliability. Just as a four-engine airplane is more likely to experience an engine failure in a given period than a craft with two of the equivalent engines, so too is it 10 times more likely that a cluster of 10 machines will require a service call. Unfortunately, many of the components that get replicated in clusters—power supplies, disks, fans, cabling, and so on—tend to be unreliable. It is, of course, possible to make a cluster arbitrarily resistant to single-node failures, chiefly by replicating data across the nodes. Happily, there is perhaps room for some synergy here: data replicated to improve the efficiency of different kinds of analyses, as noted here, can also provide redundancy against the inevitable node failure. Once again, however, the larger the dataset, the more difficult it is to maintain multiple copies of the data.

A Meta-Definition

I have tried here to provide an overview of a few of the issues that can arise when analyzing big data: the inability of many off-the-shelf packages to scale to large problems; the paramount importance of avoiding suboptimal access patterns as the bulk of processing moves down the storage hierarchy; and replication of data for storage and efficiency in distributed processing. I have not yet answered the question I opened with: What is "big data," anyway?

I will take a stab at a meta-definition: big data should be defined at any point in time as "data whose size forces us to look beyond the tried-and-true methods that are prevalent at that time." In the early 1980s, it was a dataset that was so large that a robotic "tape monkey" was required to swap thousands of tapes in and out. In the 1990s, perhaps, it was any data that transcended the bounds of Microsoft Excel and a desktop PC, requiring serious software on Unix workstations to analyze. Nowadays, it may mean data that is too large to be placed in a relational database and analyzed with the help of a desktop statistics/visualization package—data, perhaps, whose analysis requires massively parallel software running on tens, hundreds, or even thousands of servers.

In any case, as analyses of ever-larger datasets become routine, the definition will continue to shift, but one thing will remain constant: success at the leading edge will be achieved by those developers who can look past the standard, off-the-shelf techniques and understand the true nature of the hardware resources and the full panoply of algorithms that are available to them.

Related articles on queue.acm.org

Flash Storage Today
Adam Leventhal
http://queue.acm.org/detail.cfm?id=1413262

A Call to Arms
Jim Gray
http://queue.acm.org/detail.cfm?id=1059805

You Don't Know Jack about Disks
Dave Anderson
http://queue.acm.org/detail.cfm?id=864058

References

1. Codd, E.F. A relational model for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 377–387.
2. IBM 3850 Mass Storage System; http://www.columbia.edu/acis/history/mss.html.
3. IBM Archives: IBM 3380 direct access storage device; http://www-03.ibm.com/ibm/history/exhibits/storage/storage_3380.html.
4. Kimball, R. *The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses.* John Wiley & Sons, NY, 1996.
5. Litke, A.M. What does the eye tell the brain? Development of a system for the large-scale recording of retinal output activity. *IEEE Transactions on Nuclear Science 51*, 4 (2004), 1434–1440.
6. PostgreSQL: The world's most advanced open source database; http://www.postgresql.org.
7. The R Project for Statistical Computing; http://www.r-project.org.
8. Sloan Digital Sky Survey; http://www.sdss.org.
9. Throughput and Interface Performance. Tom's Winter 2008 Hard Drive Guide; http://www.tomshardware.com/reviews/hdd-terabyte-1tb,2077-11.html.
10. WLCG (Worldwide LHC Computing Grid); http://lcg.web.cern.ch/LCG/public/.
11. Zero-One-Infinity Rule; http://www.catb.org/~esr/jargon/html/Z/Zero-One-Infinity-Rule.html.

Adam Jacobs is senior software engineer at 1010data Inc., where, among other roles, he leads the continuing development of Tenbase, the company's ultra-high-performance analytical database engine. He has more than 10 years of experience with distributed processing of big datasets, starting in his earlier career as a computational neuroscientist at Weill Medical College of Cornell University (where he holds the position of Visiting Fellow) and at UCLA.

DOI:10.1145/1536616.1536634

Article development led by acmqueue
queue.acm.org

To shield the browser from attacks, Google Chrome developers eyed three key problems.

BY CHARLES REIS, ADAM BARTH, AND CARLOS PIZANO

Browser Security: Lessons from Google Chrome

THE WEB HAS become one of the primary ways people interact with their computers, connecting people with a diverse landscape of content, services, and applications. Users can find new and interesting content on the Web easily, but this presents a security challenge: malicious Web site operators can attack users through their Web browsers. Browsers face the challenge of keeping their users safe while providing a rich platform for Web applications.

Browsers are an appealing target for attackers because they have a large and complex trusted computing base with a wide network-visible interface. Historically, every browser at some point has contained a bug that let a malicious Web site operator circumvent the browser's security policy and compromise the user's computer. Even after these vulnerabilities are patched, many users continue to run older, vulnerable versions.[5] When these users visit malicious Web sites, they run the risk of having their computers compromised.

Generally speaking, the danger posed to users comes from three factors, and browser vendors can help keep their users safe by addressing each of these factors:

▸ *The severity of vulnerabilities.* By sandboxing their rendering engine, browsers can reduce the severity of vulnerabilities. Sandboxes limit the damage that can be caused by an attacker who exploits a vulnerability in the rendering engine.

▸ *The window of vulnerability.* Browsers can reduce this window by improving the user experience for installing browser updates, thus minimizing the number of users running old versions that lack security patches.

▸ *The frequency of exposure.* By warning users before they visit known malicious sites, browsers can reduce the frequency with which users interact with malicious content.

Each of these mitigations, on its own, improves security. Taken together, the benefits multiply and help keep users safe on today's Web.

In this article, we discuss how our team used these techniques to improve security in Google Chrome. We hope our firsthand experience will shed light on key security issues relevant to all browser developers.

Reducing Vulnerability Severity

In an ideal world, all software, including browsers, would be bug-free and lack exploitable vulnerabilities. Unfortunately, every large piece of software contains bugs. Given this reality, we can hope to reduce the severity of vulnerabilities by isolating a browser's complex components and reducing their privileges.

Google Chrome incorporates several layers of defenses to protect the user from bugs, as shown in Figure 1. Web content itself is run within a JavaScript virtual machine, which acts as one form of a sandbox and protects different Web sites from each other. We use exploit barriers, such as address-space layout randomization, to make it more difficult to exploit vulnerabilities in the JavaScript sandbox. We then use a sandbox at the operating-system level to limit the process itself from causing damage, even if exploits escape the earlier security mechanisms. Here, we discuss in more detail how these layers of defense are used.

Security Architecture. Google Chrome uses a modular architecture that places the complex rendering engine in a low-privilege sandbox, which we discuss in depth in a separate report.[1] Google Chrome has two major components that run in different operating-system processes: a high-privilege browser kernel and a low-privilege rendering engine. The browser kernel acts with the user's authority and is responsible for drawing the user interface, storing the cookie and history databases, and providing network access. The rendering engine acts on behalf of the Web principal and is not trusted to interact with the user's file system. The rendering engine parses HTML, executes JavaScript, decodes images, paints to an off-screen buffer, and performs other tasks necessary for rendering Web pages.

To mitigate vulnerabilities in the rendering engine, Google Chrome runs rendering-engine processes inside a restrictive operating-system-level sandbox (see Figure 1). The sandbox aims to prevent the rendering engine from interacting with other processes and the user's operating system, except by exchanging messages with the browser kernel via an IPC channel. All HTTP traffic, rendered pages, and user input events are exchanged via such messages.

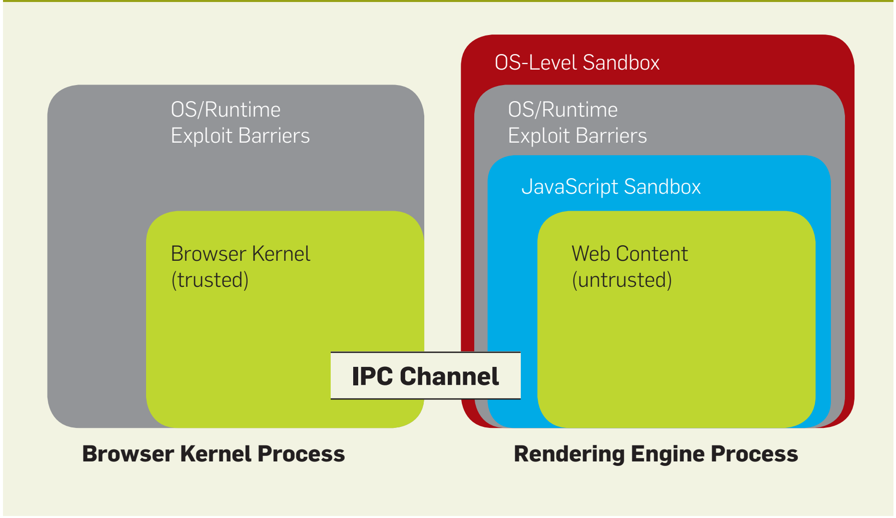
To prevent the rendering engine from interacting with the operating system directly, our Windows implementation of the sandbox runs with a restricted Windows security token, a separate and invisible Windows desktop, and a restricted Windows job object.[12] These security mechanisms block access to any files, devices, and other resources on the user's computer. Even if an attacker is able to exploit a vulnerability and run arbitrary code in the rendering engine, the sandbox will frustrate the attacker's attempts to install malware on the user's computer or to read sensitive files from the user's hard drive. The attacker's code could send messages to the browser kernel via the IPC channel, but we aim to keep this interface simple and restricted.

Getting existing code bases such as rendering engines to work fully within this type of sandbox sometimes presents engineering challenges. For example, the rendering engine typically loads font files directly from the system's font directory, but our sandbox does not allow such file access. Fortunately, Windows maintains a system-wide memory cache of loaded fonts. We can thus load any desired fonts in the browser-kernel process, outside the sandbox, and the rendering-engine process is then able to access them from the cache.

There are a number of other techniques for sandboxing operating-system processes that we could have used in place of our current sandbox. For example, Internet Explorer 7 uses a "low rights" mode that aims to block unwanted writes to the file system.[4] Other techniques include system-call interposition (as seen recently in Xax[2]) or binary rewriting (as seen in Native Client[14]). Mac OS X has an operating system-provided sandbox, and Linux processes can be sandboxed using AppArmor and other techniques. For Windows, we chose our current sandbox because it is a mature technology that aims to provide both confidentiality and integrity for the user's resources. As we port Google Chrome to other platforms such as Mac and Linux, we expect to use a number of different sandboxing techniques but keep the same security architecture.

Exploit Mitigation. Google Chrome

Figure 1. Layers of defense around Google Chrome's rendering engine.



also makes vulnerabilities more difficult to exploit by using several barriers recommended for Windows programs.[8] These include DEP (data execution prevention), ASLR (address space layout randomization), SafeSEH (safe exception handlers), heap corruption detection, and stack overrun detection (GS). These are available in recent versions of Windows, and several browsers have adopted them to thwart exploits.

These barriers make it more difficult for attackers to jump to their desired malicious code when trying to exploit a vulnerability. For example, DEP uses hardware and operating-system support to mark memory pages as NX (non-executable). The CPU enforces this on each instruction that it fetches, generating a trap if the instruction belongs to an NX page. Stack pages can be marked as NX, which can prevent stack overflow attacks from running malicious instructions placed in the compromised stack region. DEP can be used for other areas such as heaps and the environment block as well.

GS is a compiler option that inserts a special canary value into each stack call between the current top of the stack and the last return address. Before each return instruction, the compiler inserts a check for the correct canary value. Since many stack-overflow attacks attempt to overwrite the return address, they also likely overwrite the canary value. The attacker cannot easily guess the canary value, so the inserted check will usually catch the attack and terminate the process.

Sophisticated attacks may try to bypass DEP and GS barriers using known values at predictable addresses in the memory space of all processes. ASLR, which is available in Windows Vista and Windows 7, combats this by randomizing the location of key system components that are mapped into nearly every process.

When used properly, these mechanisms can help prevent attackers from running arbitrary code, even if they can exploit vulnerabilities. We recommend that all browsers (and, in fact, all programs) adopt these mitigations because they can be applied without major architectural changes.

Every large piece of software contains bugs. Given this reality, we can hope to reduce the severity of vulnerabilities by isolating a browser's complex components and reducing their privileges.

Compatibility Challenges. One of the major challenges for implementing a security architecture with defense in-depth is maintaining compatibility with existing Web content. People are unlikely to use a browser that is incompatible with their favorite Web sites, negating whatever security benefit might have been obtained by breaking compatibility. For example, Google Chrome must support plug-ins such as Flash Player and Silverlight so users can visit popular Web sites such as YouTube. These plug-ins are not designed to run in a sandbox, however, and they expect direct access to the underlying operating system. This allows them to implement features such as full-screen video chat with access to the entire screen, the user's Web cam, and microphone. Google Chrome does not currently run these plug-ins in a sandbox, instead relying on their respective vendors to maintain their own security.

Compatibility challenges also exist for using the browser's architecture to enforce the same-origin policy, which isolates Web sites from each other. Google Chrome generally places pages from different Web sites into different rendering-engine processes,[11] but it can be difficult to do this in all cases, as is necessary for security. For example, some frames may need to be rendered in different processes from their parent page, and some JavaScript calls need to be made between pages from different origins. For now, Google Chrome sometimes places pages from different origins in the same process. Also, each rendering-engine process has access to all of the user's cookies, because a page from one origin can request images, scripts, and other objects from different origins, each of which may have associated cookies. As a result, we do not yet rely on Google Chrome's architecture to enforce the same-origin policy.

Recently, some researchers have experimented with browsers (such as OP[7] and Gazelle[13]) that do attempt to enforce the same-origin policy by separating different origins into different processes and mediating their interaction. This is an exciting area of research, but challenges remain that need to be overcome before these designs are sufficiently compatible with the Web. For example, supporting existing plug-ins and communication between pages is not always straightforward in these proposals. As these isolation techniques improve, all browsers will benefit.

Reducing the Window of Vulnerability

Even after we have reduced the severity of vulnerabilities, an exploit can still cause users harm. For example, a bug might let a malicious Web-site operator circumvent the same-origin policy and read information from other Web sites (such as email). To reduce the danger to users, Google Chrome aims to minimize the length of time that users run unpatched versions of the browser. We pursue this goal by automating our quality assurance process and updating users with minimal disruption to their experience.

Automated Testing. After a vulnerability is discovered, the Google Chrome team goes through a three-step process before shipping a security patch to users:

1. The on-duty security sheriff triages the severity of the vulnerability and assigns an engineer to resolve the issue.
2. The engineer diagnoses the root-cause of the vulnerability and writes a patch to fix the bug. Often security patches are as simple as adding a missing bounds check, but other patches can require more extensive surgery.
3. The patched binary goes through a quality assurance process to ensure that the issue is actually fixed; and the patch has not broken other functionality.

For a software system as complex as a Web browser, step 3 is often a bottleneck in responding to security issues, because testing for regressions requires ensuring that every browser feature is functioning properly.

The Google Chrome team has put significant effort into automating step 3 as much as possible. The team has inherited more than 10,000 tests from the WebKit project that ensure the Web platform features are working properly. These tests, along with thousands of other tests for browser-level features, are run after every change to the browser's source code.

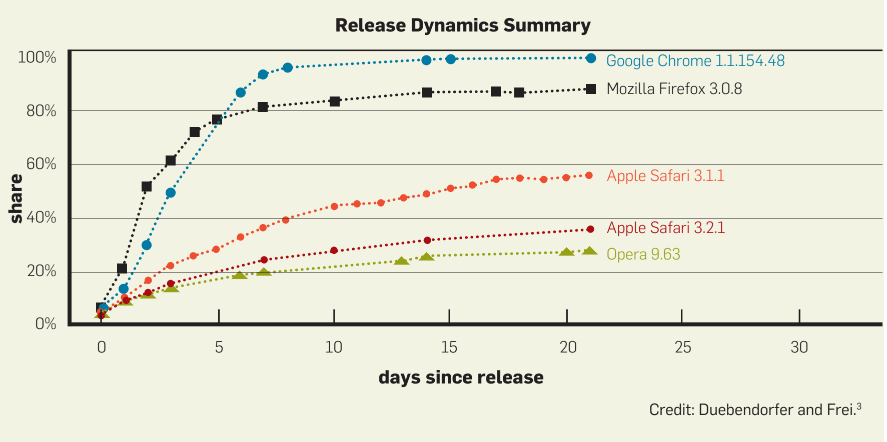
In addition to these regression tests, browser builds are tested on one million Web sites in a virtual-machine farm called ChromeBot. ChromeBot monitors the rendering of these sites for memory errors, crashes, and hangs. Running a browser build through ChromeBot often exposes subtle race conditions and other low-probability events before shipping the build to users.

Security Updates. Once a build has been qualified for shipping to users, the team is still faced with the challenge of updating users of older versions. In addition to the technical challenge of shipping updated bits to every user, the major challenge in an effective update process is the end-user experience. If the update process is too disruptive, users will defer installing updates and continue to use insecure versions.[5]

Google Chrome uses a recently open-sourced system called Omaha to distribute updates.[6] Omaha automatically checks for software updates every five hours. When a new update is available, a fraction of clients are told about it, based on a probability set by the team. This probability lets the team verify the quality of the release before informing all clients. When a client is informed of an update, it downloads and installs the updated binary in a parallel directory to the current binary. The next time the user runs the browser, the older version defers to the newer version.

This update process is similar to that for Web applications. The user's experience is never disrupted, and the user never has to wait for a progress bar before using the browser. In practice, this approach has proven effective for keeping users up to date. A recent study of HTTP User-Agent headers in Google's anonymized logs reveals how quickly users adopt patched versions of various browsers.[3] We reproduce their results in Figure 2. In these measurements, Google Chrome's auto-update mechanism updates the vast majority of its users in the shortest amount of time, as compared with other browsers. (Internet Explorer is not included in these results because its minor version numbers are not reported in the User-Agent header.)

Figure 2. Auto-update mechanisms in Google Chrome.



Credit: Duebendorfer and Frei.[3]

Reducing Frequency of Exposure

Even with a hardened security architecture and a small window of vulnerability, users face risks from malicious Web site operators. In some cases, the browser discourages users from visiting known malicious Web sites by warning users before rendering malicious content. Google Chrome and other browsers have taken this approach, displaying warning pages if a user tries to visit content that has been reported to contain malware or phishing attempts. Google works with StopBadware.org to maintain an up-to-date database of such sites, which can be used by all browsers.

One challenge with using such a database is protecting privacy. Users do not want every URL they visit reported to a centralized service. Instead, the browser periodically downloads an efficient list of URL hashes without querying the service directly. To reduce the space required, only 32-bit prefixes of the 256-bit URL hashes are downloaded. This list is compared against a list of malicious sites. If a match is found for a prefix, the browser queries the service for the full 256-bit hashes for that prefix to perform a full comparison.

Another challenge is minimizing false positives. Google and StopBadware.org have tools to help publishers remove their pages from the database if they have been cleaned after hosting malware. It is also possible for human errors to flag sites incorrectly, as in an incident in January 2009 that flagged all URLs as dangerous.[9] Such errors are typically fixed quickly, though, and safeguards can be added to prevent them from recurring.

These services also have false negatives, because not every malicious page on the Web can be cataloged at every point in time. Although Google and StopBadware.org attempt to identify as many malicious pages as possible,[10] it is unlikely to be a complete list. Still, these blacklists help protect users from attack.

Conclusion

There is no silver bullet for providing a perfectly secure browser, but there are several techniques that browser developers can use to help protect users. Each of these techniques has its own set of challenges.

In particular, browsers should minimize the danger that users face using three techniques:

- Reduce attack severity by applying the principle of least privilege in the browser architecture. This technique limits the damage caused when an attacker exploits a vulnerability.
- Reduce the window of vulnerability by ensuring updates are developed and deployed as quickly as possible. This technique minimizes the number of vulnerable browsers an attacker can target.
- Reduce how often users are exposed to attacks by filtering out known malicious content. This technique protects users during vulnerable time windows.

The Google Chrome team has focused on each of these factors to help provide a secure browser while preserving compatibility with existing Web content. To make Google Chrome even more secure, we are investigating further improvements to the browser's security architecture, such as mitigating the damage that plug-in exploits can cause and more thoroughly isolating different Web sites using separate sandboxed processes. Ultimately, our goal is to raise the bar high enough to deter attackers from targeting the browser. **C**

There is no silver bullet for providing a perfectly secure browser, but there are several techniques that browser developers can use to help protect users.

Related articles on queue.acm.org

Security in the Browser
Thomas Wadlow and Vlad Gorelik
http://queue.acm.org/detail.cfm?id=1516164

Cybercrime 2.0: When the Cloud Turns Dark
Niels Provos, Moheeb Abu Rajab, and Panayiotis Mavrommatis
http://queue.acm.org/detail.cfm?id=1483106

Phishing for Solutions
Kode Vicious
http://queue.acm.org/detail.cfm?id=1142063

References

1. Barth, A., Jackson, C., Reis, C., and Google Chrome Team. The Security Architecture of the Chromium Browser (2008); http://crypto.stanford.edu/websec/chromium/chromium-security-architecture.pdf.
2. Douceur, J.R., Elson, J., Howell, J., and Lorch, J.R. Leveraging legacy code to deploy desktop applications on the Web. In *Proceedings of Operating Systems Design and Implementation* (2008).
3. Duebendorfer, T., Frei, S. Why silent updates boost security. ETH Tech Report TIK 302 (2009); http://www.techzoom.net/silent-updates.
4. Franco, R. Clarifying low-rights IE. IEBlog (June 2005); http://blogs.msdn.com/ie/archive/2005/06/09/427410.aspx.
5. Frei, S., Duebendorfer, T., and Plattner, B. Firefox (in) security update dynamics exposed. *ACM SIGCOMM Computer Communication Review 39*, 1 (2009).
6. Google. Omaha: Software installer and auto-updater for Windows. Google Code; http://code.google.com/p/omaha/.
7. Grier, C., Tang, S., and King, S.T. Secure Web browsing with the OP Web browser. In *Proceedings of IEEE Symposium on Security and Privacy* (2008).
8. Howard, M., Thomlinson, M. Windows Vista ISV Security (2007); http://msdn.microsoft.com/en-us/library/bb430720.aspx.
9. Mayer, M. "This site may harm your computer" on every search result. The Official Google Blog (Jan. 2009); http://googleblog.blogspot.com/2009/01/this-site-may-harm-your-computer-on.html.
10. Provos, N., McNamee, D., Mavrommatis, P., Wang, K., and Modadugu, N. The ghost in the browser: Analysis of Web-based malware. In *Proceedings of the First Usenix Workshop on Hot Topics in Botnets* (April 2007).
11. Reis, C. and Gribble, S.D. Isolating Web programs in modern browser architectures. In *Proceedings of European Conference on Computer Systems* (April 2009).
12. Sandbox. Chromium Developer Documentation (2008); http://dev.chromium.org/developers/design-documents/sandbox.
13. Wang, H.J., Grier, C., Moshchuk, A., King, S.T., Choudhury, P., and Venter, H. The Multi-Principal OS Construction of the Gazelle Web Browser. Microsoft Research Technical Report (MSR-TR-2009-16) 2009; http://research.microsoft.com/pubs/79655/gazelle.pdf.
14. Yee, B., Sehr, D., Dardyk, G., Chen, J. B., Muth, R., Ormandy, T., Okasaka, S., Narul, N., and Fullagar, N. Native Client: A sandbox for portable, untrusted x86 native code. In *Proceedings of IEEE Symposium on Security and Privacy* (2009)

Charles Reis is a software engineer at Google working on the Google Chrome Web browser. He recently completed his Ph.D. in the Department of Computer Science and Engineering at the University of Washington. His research focuses on improving the reliability and security of Web browsers and Web content.

Adam Barth is a postdoctoral fellow at the University of California, Berkeley. His research focuses on the security of modern Web browsers, including their security policies, enforcement mechanisms, and security user interfaces. He is a contributor to the Chromium, WebKit, and Firefox open source projects and is an invited expert to the W3C HTML and Web Applications working groups.

Carlos Pizano is a senior software engineer at Google working on the Google Chrome Web browser. His work focuses on security and sandboxing for Internet-facing applications.

DOI:10.1145/1536616.1536633

Article development led by acmqueue
queue.acm.org

The age of cloud computing has begun. How can companies take advantage of the new opportunities it provides?

BY MACHE CREEGER

CTO Roundtable: Cloud Computing

MANY PEOPLE READING about cloud computing in the trade journals will think it's a panacea for all their IT problems—it is not. In this CTO Roundtable discussion we hope to give practitioners useful advice on how to evaluate cloud computing for their organizations. Our focus will be on the SMB (small- to medium-size business) IT managers who are underfunded, overworked, and have lots of assets tied up in out-of-date hardware and software. To what extent can cloud computing solve their problems? With the help of five current thought leaders in this quickly evolving field, we offer some answers to that question. We explore some of the basic principles behind cloud computing and highlight some of the key issues and opportunities that arise when computing moves from in-house to the cloud. Our sincere thanks to all who participated in the roundtable, and to the ACM Professions Board for making this event possible.

Participants

Werner Vogels is the CTO of Amazon.com, responsible for both e-commerce operations and Web services. Prior to working for Amazon he was a research scientist at Cornell University, studying large, reliable systems.

Greg Olsen is the CTO and Founder of Coghead, a platform-as-a-service (PaaS) vendor on both sides of the cloud equation. Coghead sells cloud-based computing services as an alternative to desktop or client/server platforms and is also a consumer of cloud services. The company built its entire service on top of Amazon's Elastic Compute Cloud (EC2), Elastic Block Storage (EBS), and Simple Storage Service (S3). Previously, Olsen founded Extricity, a company that provided business-to-business integration.

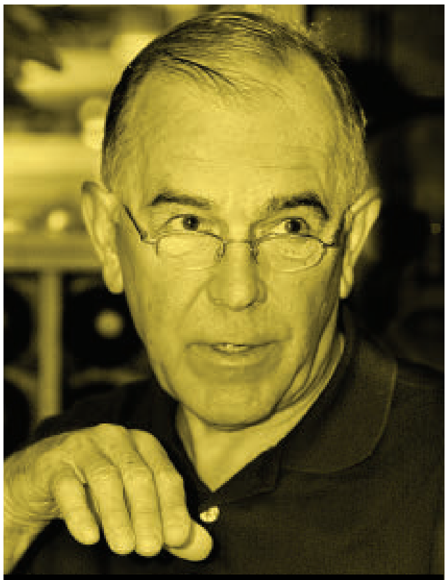
Lew Tucker is CTO of cloud computing at Sun Microsystems. In the 1980s he worked on the Connection Machine, a massively parallel supercomputer that sparked his interest in very large-scale computing. He spent 10 years at Sun as VP of Internet services running Sun's popular Web sites. Tucker left Sun to go to Salesforce.com, where he created AppExchange (http://www.salesforce.com/appexchange/), and afterward went to a start-up called Radar Networks. Recently he returned to Sun to lead its initiative in cloud computing.

Greg Badros is senior engineering director at Google, where he has worked for six years. Before that he was chief architect at Infospace and Go2Net. He earned his Ph.D. in constraint algorithms and user experiences from the University of Washington.

Geir Ramleth is CIO of Bechtel, where he provides cloud services for internal company use. Prior to his current

Mache Creeger

Lew Tucker


Steve Bourne

The Group

Geir Ramleth

Werner Vogels

Greg Badros

Greg Olsen

PHOTOGRAPHS BY MARJAN SADOUGHI

job, Ramleth started a company inside Bechtel called Genuity, which was an early ISP and hosting company. Genuity was later sold to GTE.

Steve Bourne is CTO at El Dorado Ventures, where he helps assess venture-capital investment opportunities. Prior to El Dorado, Bourne worked in software engineering management at Cisco, Sun, DEC, and Silicon Graphics. He is a past president of ACM and chairs both the ACM Professions Board and the ACM Queue Editorial Board.

Moderator

Mache Creeger is principal of Emergent Technology Associates, where he provides marketing and business development enterprise infrastructure consulting for large and small technology companies. Beginning his career as a research computer scientist, Creeger has held marketing and business development roles at MIPS, Sun, Sony, and InstallShield, as well as various startups. He is an ACM columnist and moderator and head wrangler of the ACM CTO Roundtable series.

CREEGER: Let's begin the discussion with a general question and then dig down into some of the deeper issues. How would you define cloud computing?

TUCKER: Cloud computing is not so much a definition of a single term as a trend in service delivery taking place today. It's the movement of application services onto the Internet and the increased use of the Internet to access a wide variety of services traditionally originating from within a company's data center.

BADROS: There are two parts to it. The first is about just getting the computation cycles outside of your walled garden and being able to avoid building data centers on your premises.

But there's a second aspect that is equally important. It is about the data being in the cloud and about the people living their lives up there in a way that facilitates both easy information exchange and easy data analysis.

The great search tools available today are a direct result of easy access to data because the Web is already in the cloud. As more and more user data is stored in the cloud, there is a huge opportunity that transcends just computation being off-premises because there is a relatively high-bandwidth connection to all those bits.

“

LEW TUCKER

Cloud computing is not so much a definition of a single term as a trend in service delivery. It's the movement of application services onto the Internet and the increased use of the Internet to access a variety of services traditionally originating from within a company's data center.

”

TUCKER: Tim O'Reilly's definition of Web 2.0 was that the value of data significantly increases when a larger community of people contributes. Greg [Badros]'s characterization complements that nicely.

VOGELS: It's not just data. I also believe that clouds are a platform for general computation and/or services. While telcos are moving their platforms into clouds for cost-effectiveness, they also see opportunities to become a public garden platform. In this scenario, people can run services that either extend the telco's services or operate independently. If, for example, you want to build an application that has click-to-call or a new set of algorithms such as noise detection in conference calls, then you can run those services connecting to the telco's platform. The key is having execution access to a common platform.

Because we have a shared platform, we can do lots of new things with data, but I believe we can do new things with services as well.

TUCKER: I see it as three layers: SaaS (software-as-a-service), which delivers applications such as Google Apps and Salesforce.com; PaaS (platform-as-a-service), which provides foundational elements for developing new applications; and IaaS (infrastructure-as-a-service), which is what Amazon has led with, showing that infrastructure can also be accessed through the cloud. I believe it is in this infrastructure layer—in which we've virtualized the base components of compute and storage, delivering them over the Internet—where we have seen the fundamental breakthrough over the past two years.

VOGELS: Understanding cloud computing requires a look at its precursors, such as SaaS before it became this platform-like environment; SOA (service-oriented architecture); virtualization (not just CPU virtualization but virtualization in general); and massively scalable distributed computing.

These were technologies that we needed to understand fully before cloud computing became viable. We needed to be able to provide these services at scale, in a reliable manner, in a way that only academics thought about 10 years ago. Building on this foundation, we have now turned these precursors into the commercial practice of cloud computing.

TUCKER: A handful of companies, such as Amazon, Google, and Yahoo, demonstrated the advantage of very, very large scale by building specialized architectures just to support a single application. We have started to see the rest of the world react and say, "Why can't we do that?"

BADROS: While I agree that the emergence of the massive scale of these companies plays a critical part, I also think that the development of client-side technology such as HTML, CSS, AJAX, and broadband connectivity is very important.

CREEGER: What about virtualization? It provides an encapsulation of application and operating system in a nice, neat, clean ABI (application binary interface). You could take this object and put it on your own premises-based hardware or execute it on whatever platform you choose. Virtualization makes execution platforms generic by not requiring the integration of all those horrible loose ends between the application and the operating system every time you want to move to a new machine. All that is required for a virtualized application/operating-system pair to execute on a new platform is for that platform to support the VM (virtual machine) runtime.

TUCKER: An important shift has been to use basic HTTP, in the form of REST (representational state transfer) APIs (http://en.wikipedia.org/wiki/Representational_State_Transfer), as an easier-to-use SOA framework. Everything that made services hard before, such as CORBA (http://www.omg.org/gettingstarted/corbafaq.htm) or IDL (http://en.wikipedia.org/wiki/Interface_description_language), went away when we said, "Let's do it all over HTTP."

BOURNE: Let's be practical. What are the economics of clouds? What's the CapEx (capital expenditure) and what is the OpEx (operational expenditure)? At the end of the year, did I spend more or less?

VOGELS: CapEx forces you to make massive investments. In the past, you had some measure of control over your customers; these days your customers have control over you. They know what to choose and have perfect information. So if you build products today as an enterprise, but also as a young business, you have no idea whether you're going to be successful or not. The less investment you have to make upfront, the better.

OLSEN: What inspired me about the cloud was that I could start a company and not buy any servers, phones, or software licenses. We were dedicated to using cloud services from day one. We started our company relying solely on services for email and the Internet and went from there to putting our source control on as a service. I wrote an article titled "Going Bedouin" where I expressed these views in more detail (http://webworkerdaily.com/2006/09/04/going-bedouin/).

BADROS: Clouds are clearly a huge win to get started with a business or product offering. At Google, we see internal people using the GAE (Google App Engine; http://code.google.com/appengine/) as a means of deploying something very quickly before they worry about scaling it on our base infrastructure. People do this because it is so much faster to get going, even inside Google where you have lots of infrastructure available.

Today's developer has a decision to make: after I am a success, am I going to switch off of this initial platform? That's the trade-off. Once it's obvious that something like an Amazon S3 is able to outperform the best that the vast majority of companies can ever deploy, then it's obvious you should just work entirely within the cloud. In this way you never have to suffer the replacement CapEx for the initial infrastructure.

VOGELS: For many customers, using our cloud products requires new expertise. You are no longer looking for a typical system administrator. If you have a large company, you're looking for someone with the specific expertise to support 50,000 internal customers. Using the cloud, you no longer have to deal with things at the physical level. You no longer need to have people running around the data center replacing disks all day.

RAMLETH: You can get your smartest guys to work on what matters most rather than having them work on mundane stuff. That's a huge benefit.

CREEGER: What about the people who need to run a flat-load, basic accounts receivable package? Once they get their software and hardware in place and get their operational process down, it's pretty straightforward and they can amortize the capital expenditure over a very long time period.

TUCKER: Every three years they've got to upgrade the software and the hardware.

RAMLETH: We spent $5 million last year on an upgrade that did nothing for our business processes or end users. The software vendors told us that if we did not upgrade, they would stop supporting us.

OLSEN: I always wondered why we think software is so different from anything else. If a restaurant was growing its own food, slaughtering its own animals, generating its own power, collecting rainwater, and processing its own sewage, we would all think they were idiots for not using ready-made services. For a long time people built their own stack from the ground up, or ran their own servers because they could. Viewing the state of our industry, any student of economics will tell you that you have to start layering.

VOGELS: There are restaurants that do not buy their own herbs; they grow them on-site. They would argue that it contributes to the quality of the end product. They will never generate their own electricity, however, because that will not produce better food.

OLSEN: Realistically, however, software is really extreme in terms of how many people are doing undifferentiated tasks, on their own, at all kinds of levels. Look at the auto industry: there are many tiers of subcontractors, each providing specialized services and products. We just haven't evolved to that same level of efficiency.

RAMLETH: We have dramatically reduced our data-center capital expenditures as a direct result of virtualization, allowing us to reuse our capital many more times than we ever could before. Before we started our effort, the average server utilization in our global server park was 2.3%. Going to virtualization has increased it to between 60% and 80%.

When we started, the core side of our central data centers, not including peripheral things, ran 35,000 square feet. Today, the equivalent of those 35,000 square feet is now operating in less than 1,000 square feet. We are utilizing our hardware in very different ways than we could ever do before. The lesson we learned is that a very big part of building these public and private clouds is to be sure that you can get utilization factors significantly better than traditional company operations.

VOGELS: If you run your services inside the company, privately, utilization becomes an issue. It amortizes your costs over a number of cycles. If you run services outside, on a public service, it is no longer an issue for you.

RAMLETH: We are operating hundreds of servers that are processing data for projects that no longer exist and are no longer generating revenue. We do this because there may be a time and place where we would need this information, such as in a warranty situation.

Amazon taught us that we can move these programs from our data center to EC2, get them operational, capture that image, and then shut it down. At this point we have incurred very minimal costs. When conditions arise that require the execution of one of those programs, we can do it. By using Amazon EC2, we can transform what used to be a fixed cost of allocating a dedicated in-house server—regardless of whether we need the information—to a variable cost that is incurred only when the business case requires it.

The cost savings of using Amazon is quite compelling. A basic server, operating internally, that sits and does nothing costs us about $800 to a $1,000 per month to run. We can go to Amazon and get charged only for what we use, at a rate of 10 cents to 15 cents an hour.

TUCKER: This is the promise of utility computing. Users will be able to move their applications and their platforms off-site, and they will have more choices. There will be many different kinds of cloud service providers and, ultimately, opportunities for arbitrage. We are moving to a scenario where it will not matter where things execute, and where choosing an execution platform will be based on a number of different characteristics such as cost, security, performance, reliability, and brand awareness.

The great thing is that self-service has now moved into the provisioning of virtualized compute, storage, and networking resources. Without even talking to anybody at Amazon, you can use its service with just a credit card. Enterprise customers are looking at their internal customers the same way. If the marketing department wants to run a new kind of application, traditionally you had to get the IT department to agree to help you build and deploy that application. Now IT departments are able to say, "You've got your own developers over in your area. If they want to develop and run this, fine, go ahead. Here are the policies for infrastructure services."

BADROS: One of the key benefits is that not only is it easier to get going at start up, but also there is no discontinuity as things grow. It's never the case that you are debating internally whether you should buy that extra server, invest in a more sophisticated infrastructure, or be able to scale to that second machine.

TUCKER: We need to be a little careful. Not all applications scale easily. While there is a whole class of applications that have very easy scaling characteristics, others do not. Databases are part of this class unless you are using something that has been set up to scale, such as Amazon's SimpleDB (http://aws.amazon.com/simpledb/). If you're running your own database, unless it has been designed to be scalable, don't count on it happening.

CREEGER: How does that poor person sitting at a small- to mid-cap company make a decision to invest in clouds? What is he going to do next quarter or next year when the CEO comes in and says, "I read this thing in the *Wall Street Journal* stating that all the smart companies are going to cloud computing." How is this guy going to respond?

OLSEN: First, adopt a philosophy of buy first, build second—even at the basic level of "I'm going to start a company, I need IT services." Do I look to hire engineers and buy equipment or do I assume that there's some outside service that might meet my needs? To me that's half of it. I'm going to assume that services meeting my needs are already available or are going to evolve over time. I take a philosophy that says, "I'm all about my core business. I buy only infrastructure that directly supports my unique contributions to the marketplace."

RAMLETH: I agree with you, if you are rational. However, you're dealing with humans, and they are often not rational. When a CEO goes down to his IT manager's office and asks, "How are we utilizing cloud computing?", the first thing that manager asks is, "What will this mean to me?" The biggest obstacle to change at our company was our own IT guys trying to protect their jobs. The change we have done at Bechtel has been 20% technology and 80% managing the change.

I believe an important part of your value proposition should be to explain to both the decision maker as well as the user how this tool enhances their professional futures. If it does not, those folks are going to be your obstacles.

TUCKER: There are certainly different approaches for different businesses at different points in their life cycles. A start-up has a certain set of needs. I completely agree with Greg [Olsen] to look for all the services that you can purchase before you think of building it yourself.

Animoto is a new company that makes movies out of photographs synced with music. It started with 50 instances running on Amazon. They launched it on Facebook and had very high success. In a matter of three days they went to 3,500 instances.

Can you imagine going to your IT department and saying, "We're running on 50 servers today, and in two to three days we want to go to 3,500 servers"? It just would not have been possible.

CREEGER: So, for the zero- to a million-miles-an-hour overnight business plan that is stalled because of up-front CapEx costs, cloud computing is going to be your answer.

What other types of criteria can we give to people to evaluate how effective their internal IT infrastructure is in supporting business goals?

VOGELS: There are many first steps that corporations take into this world. Engineers can start by experimenting with these services, using them for small projects and comparing cost savings. I find that many of the first steps that enterprises take are just something small, easy, simple, and cost effective.

The *New York Times* scanned images covering a 60-year period in history and wanted to place them online. These guys moved four terabytes into S3, ran all the stuff on a Sunday, spent $25, and got the product done.

BADROS: Replacing existing organization structure or IT functionality is harder in larger companies. Often you have a better chance of success if you introduce something that provides new value, perhaps by enabling a new type of collaboration, rather than replacing or modifying existing functionality. In this way you can avoid the risk of encountering resistance resulting from complexity or politics. In today's tougher economic times, you may also want to make your proposal more compelling by showing that operational TCO (total cost of ownership) can be significantly lowered when using a cloud.

OLSEN: The assumption that it's central IT making decisions about other technologies is wrong. Cloud computing has become successful not because a whole bunch of central IT groups proclaimed that cloud computing is good. Cloud computing has become popular from grassroots acceptance, from IT decisions made by small businesses, new providers, or at the departmental level. Cloud computing is coming into IT only at the end of this. My company does not sell to CIOs. We don't even try.

CREEGER: That's fine, but there are CIOs who will have to provide plans after their CEOs read that one can realize massive savings with cloud computing.

BOURNE: So who should pay attention to cloud computing?

OLSEN: I'm either a consumer of information technology needs: I need applications, I need storage; or I'm a producer: I'm somebody who's going to provide a service. Both of those audiences need to know what they can build from and how they can sell what they have. To me, it's not primarily about central IT. Central IT is an important constituent, but all these little system integrators, consultants, little ISVs, VARs—these are the folks who actually deploy computation on a broad scale to businesses and people. Any person who is in that space, either as a producer or a consumer of IT, needs to understand how to use cloud services.

BADROS: To me, the value proposition of cloud computing is so broad that the beauty of it is you can sell to almost anybody in the organization. Different aspects of the solution appeal to different sets of folks. Depending on whom I'm talking to, the story is different in order to let them see how it's going to be better for them.

“

WERNER VOGELS

If you run your services inside the company, privately, utilization becomes an issue. It amortizes your costs over a number of cycles. If you run services outside, on a public service, it is no longer an issue for you.

”

The individual who has been using consumer email and Google Calendar is excited about having the home experience at work and about the rich search capabilities and collaboration of Calendar. We see people using docs and spreadsheets to manage their wedding on the docs collaboration suite. Then when they are doing a similar type of project at work, they don't understand why they are stuck in early 1990s-style thinking with a set of applications that don't talk to one another. For that person, the collaboration story is the value proposition.

If an enlightened CIO comes to us and is wondering how this thing helps his organization, then cost of ownership, ease of scaling, and simplicity of starting new geographically distributed offices are really rich selling points.

To the CEO, it may be the fact that the IT department doesn't need to be as large as it is. The CEO is often scratching his head asking why he is spending 20% of his people budget just so the rest of his people can get their email. So, it really depends on the audience to understand what the best value proposition is. The beauty of cloud computing is there is a story for everyone—it's that compelling.

CREEGER: Does cloud computing enable new types of functionality that were not feasible under more traditional IT architectures?

VOGELS: In the past, I always thought that you could not build data warehouses out of general components. It's highly specialized, and I thought being really fine-grained precluded you from doing scatter-gather of lots of data operations. I think MapReduce (http://labs.google.com/papers/mapreduce-osdi04.pdf) has shown us that brute force works, and while it's not the most efficient approach, it allows you to get the job done in a very simple way.

A number of small companies now provide data warehousing as a service. The data movement is a little more inefficient than it used to be, but they're getting access to much smarter, much easier-to-use computational components.

It turns out that we have many customers who do not need a data warehouse 24-hours-a-day. They need it two hours a week. In the worst case, they're willing to spend a bit more on computational resources just to get these two hours. They are still ahead on cost, given the alternative of having to purchase the hardware outright and build it up to support a peak load.

CREEGER: So, the analogy would be to analyze the cost of either purchasing a car or taking taxis to meet personal transportation needs?

VOGELS: Engineers are not well trained to think about end-to-end cost. MapReduce and other examples have shown us that the end-to-end picture of cost looks very different from what you would normally expect. We have to learn to think about the whole package—at storage, computation, and what the application needs to do—and really reason about what the axis of scale and cost really is.

CREEGER: I'd like to go around the room once and give some final recommendations to the folks who are struggling to try to make sense of all this.

RAMLETH: This is not a technology game but a change-management game. The goal is to get people to understand that it is not dangerous to think this way. We have three rules:

- Think about what you can do that can benefit service delivery in aggregate; don't focus on the small subcomponents that can lead to suboptimal solutions.
- Don't think about how you're going to distribute your costs before you start any effort. Make sure that internal charging mechanisms (allocations) are not obstacles for change and progress.
- Don't think about and design future organization changes. Base decisions on organizational benefit and not on increased power to you as a manager or to your organization.

If you think about these three things, it's amazing what an organization can actually do.

BADROS: The beauty of what we're talking about is that it's so easy to try. You don't need a big budget or approvals to get started. The fact that you can do this so simply enables innovation that would be unavailable if you needed to purchase a big piece of hardware ahead of time.

TUCKER: As services move into the Internet, they become easier and more cost effective. This also means a shift in power in IT away from those who control capital resources to the users and developers who use self-service to provision their own applications. When FedEx went online, people were taken out of the support loop and customers could find their package status information themselves whenever it was needed. You can now apply the same principle to the provisioning of computing resources. A developer can have a server provisioned to run an application without having to contact a human. That cuts the most costly aspect of computing out of the equation.

OLSEN: Cloud computing presents a compelling opportunity for consumers of information technology and producers of information services. Application builders should take advantage of existing functionality they can buy as opposed to the past practice of building their own and focus their resources on the unique capability they alone can deliver. Consumers of information technology have got to rethink where they look for functionality. If they don't adapt their service delivery models, then they will quickly become obsolete.

CREEGER: Reducing cost and enabling overall agility are what I believe you all are trying to say. Cloud computing has the potential for removing business friction to make more services possible and to do so much more easily, with less risk and capital outlay. I think that is as good a summary as any for something as transformative as cloud computing. Thank you all very much for your time, talent, and wisdom.

> **GREG OLSEN**
> **Cloud computing presents a compelling opportunity for consumers of information technology and producers of information services.**

Related articles on queue.acm.org

For the complete version of this CTO Roundtable discussion, visit
http:// http://queue.acm.org/detail.cfm?id=1551646

Describing the Elephant
Ian Foster and Steven Tuecke
http://queue.acm.org/detail.cfm?id=1080874

Enterprise Software as Service
Dean Jacobs
http://queue.acm.org/detail.cfm?id=1080875

CTO Roundtable: Virtualization
Mache Creeger (moderator)
http://queue.acm.org/detail.cfm?id=1400229

Mache Creeger (mache@creeger.com) is a 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.

DOI:10.1145/1536616.1536636

Meaning can be as important as usability in the design of technology.

BY KRISTEN SHINOHARA AND JOSH TENENBERG

A Blind Person's Interactions with Technology

CURRENT PRACTICE IN computer interface design often takes for granted the user's sightedness. But a blind user employs a combination of other senses in accomplishing everyday tasks, such as having text read aloud or using fingers along a tactile surface to read Braille. As such, designers of assistive technologies must pay careful attention to the alternatives to sight to engage a blind user in completing tasks. It may be difficult for a sighted designer to understand how blind people mentally represent their environment or how they apply alternative options in accomplishing a task. Designers have responded to these challenges by developing alternative modes of interaction, including audible screen readers,[11] external memory aids for exploring haptic graphs,[20] non-speech sounds for navigating hypermedia,[16] two-finger haptic interfaces for touching virtual objects,[22] haptic modeling of virtual objects,[13] and multimodal (auditory, haptic, visual) feedback for simple computer-based tasks.[10] The effectiveness of these alternative modes of interaction is studied primarily through a usability framework, where blind and visually impaired users interact with specific devices in a controlled laboratory environment. These developments in assistive technology make a point to take advantage of the alternative modes of interaction available to blind users.

Physical obstacles are not the only considerations affecting interaction between blind users and everyday artifacts. As we found in this study, elements of meaning, such as socialization, efficiency, flexibility, and control, strongly influence the use of both digital and non-digital artifacts by blind users. Taken-for-granted factors, such as an individual's social ties or busy schedule, might determine whether and how an object is used. Therefore, designers may need to pay close attention to the external factors that influence an individual's choice and use of technology. Conversely, and equally as important, designers must also consider how an individual's internal values and desires affect their technology preferences.

The study described here is an in-depth exploratory and descriptive case study[24] of a blind individual using various technologies in her home. Previous studies in lab settings compared interactions against a set of heuristics or with a control group, allowing researchers to isolate events in order to understand how users interact with specific technologies on a narrow range of tasks. We took this study out of the lab and into the home to get a better sense of the nuances of everyday life influencing how a blind user interacts with technology. It differs from the usability approaches in several ways. First, we wanted to look across a range of technologies for common kinds of task fail-

BrailleNote from HumanWare; http://www.humanware.com/en-usa/home.

PHOTOGRAPH BY RUSS WEAKLEY

ure and workarounds, rather than on a single technology or task. Second, because emerging technologies involve a choice of what to place in hardware and what to place in software, such as whether to have physical or virtual buttons on a cellphone, we wanted to investigate user interaction with both digital and physical objects to better understand the trade-offs in hardware vs. software design choices. Third, the investigation was situated within the individual's home rather than in the laboratory to better understand artifact use in a naturalistic setting. And fourth, our interviews concerned not only usability but aesthetics, affect, meaning, historical associations of use in context, and envisioning of future technologies. Overall, we were concerned about what technologies were most valued and used, when they were used and for what purpose, the difficulties experienced in their use, the workarounds employed, and the meanings and interpretations associated with their use.

Without careful consideration for both the limitations in usability and the meaning of the interactions affecting blind users, sighted technology designers may unwittingly create interfaces with the wrong affordances or that are dissonant with a user's personal preferences, resulting in task failure. Already known is that the visually impaired must make alternative accommodations to accomplish the same tasks day in and day out. What is little known is how much of an influence an individual's personal values and surroundings have on the choice of where, when, and how technology is used. Observations in a user's home of interactions with existing technologies may provide insight into the way surroundings and personal preferences are drawn on to help complete daily tasks.

As we suggest in the study, the combination of functionality *and* socially situated meaning determines for the user the actual usability of a technology to accomplish specific tasks. These technologies hold meaning that affects the ways individuals understand themselves in relation to the communities to which they belong.

Background

Developing the study, we drew on a number of literatures, including in assistive technology for people with visual impairments, task breakdowns and workarounds, and design ethnography in the home:

Design ethnography. The study design reflects Clifford Geertz's view that "man is an animal suspended in webs of significance he himself has spun."[8] Significance is constructed not only from behavior and discourse, but in the materials with which people interact. Many are mundane objects—measuring cups, cellphones, sticky notes.

And yet, as Csikszentmihalyi and Rochberg-Halton[6] wrote, these objects become infused with meaning through use and association. "Humans display the intriguing characteristic of making and using objects. The things with which people interact are not simply tools for survival or for making survival easier and more comfortable. Things embody goals, make skills manifest, and shape the identities of their users. Man is not only *homo sapiens* or *homo ludens*, he is also *homo faber*, the maker and user of objects, his self to a large extent a reflection of things with which he interacts. Thus objects also make and use their makers and users."[6] If, as technology designers, we desire to improve the human condition through our intentional acts of design, then our central concern should be the ways in which technologies are woven into human webs of significance.

In order to elicit a more holistic perspective on the usability of artifacts for a blind individual, we extended the study of human-machine interaction from the workplace into the home, as have other recent researchers.[2,4] Drawing on traditional ethnographic methods used in the social sciences,[8,18] this research examines the situated, physical interactions between people and artifacts, as well as the meanings people attribute to specific technologies and the personal perspectives they bring to their interactions. As Bell et al.[2] note: "The potential situated meanings of domestic technology are fluid and multiple, connecting with a range of discourses, such as work, leisure, class, religion, age, ethnicity, sex, identity, success. Meaning may also be embodied in artifacts through the historical contexts of use."

Though undertaking this investigation from the comfort of our university lab would have been convenient for us as researchers, doing so would have undervalued the importance of place in evoking the meaning of everyday things. Homes are not just shelter, but places where people *dwell*, where one finds the "'lived relationships' that people maintain with places."[1] Our intention was that by observing and interviewing in our informant's place of dwelling, deeper associations of significance would be evoked related to the objects found there. As the ethnographer Keith Basso[1] points out, "places possess a marked capacity for triggering acts of self-reflection, inspiring thoughts about who one presently is, or memories of who one used to be, or musings on who one might become. That is not all. Place-based thoughts about the self lead commonly to thoughts of other things—other places, other people, other times, whole networks of associations that ramify unaccountably within the expanding spheres of awareness that they themselves engender."

Breakdowns and workarounds. We are interested in both the success and failure a nonsighted person experiences in interaction with technological artifacts. We are particularly interested in understanding the task failures, what Winograd and Flores[21] called "breakdowns" in that they reveal what is often invisible during successful artifact use. Task failures are unsurprising, given that many of the artifacts used daily by people who are blind have been constructed in a coevolved biological and social world in which sight is the norm. Task failures are also not failures in the sense that they are merely the stopping

Limitations and workarounds.

Object/Task	Limitation	Explanation	Workaround	Key Insight
Navigating with JAWS	JAWS does something other than the intended action.	Other keys might have been hit by mistake.	Keeps trying different key combinations; satisfactory but inefficient.	*Socialization.* Negatively affects online interactions.
Setting alarm on tactile watch	Tactile watch lacks alarm function.		Sets alarm on other electronic devices.	*Efficiency.* More efficient than using multiple other devices as alarms.
Accessing time on Talking Watch	Watch is "clunky" and obtrusive.	The talking draws unwanted attention to her from others.	Use of tactile watch is consistently successful.	*Socialization.* Does not want to call attention to herself.
Searching for a CD	She cannot quickly read CD covers; in worst case requires linear search through all CDs.	Labels do not fit on case spines.	Mentally organize by preference, read one at a time; efficient and reliable.	*Efficiency.* Search process slow but effective.
Labeling CDs for sighted friends with labeler	Labeler creates only Braille labels.	Sighted people do not know how to read Braille.	Gives unlabeled CDs to friends.	*Socialization.* Wants to create print labels for CDs for friends.
Labeling CDs for herself with labeler	Creating labels is time consuming.		Often does not label discs.	*Independence.* Labels allow control of surroundings.
Measuring water with measuring cups	Cannot see orientation of the cups.	Difficult to tell if a cup is right side up when held by flat handle.	Uses free hand to feel orientation of cup before measuring.	
Pouring water from cup into bowl	Cannot see where the bowl is in relation to the sink.	Forgets location of other utensils on countertop.	Uses free hand to keep cup level and find bowl.	
Receiving text messages by cellphone	Unable to access and read messages received.	Text messages available only via visual screen.	Finds a friend to read messages to her; inefficient and unsatisfactory.	*Control.* Prefers calling others to getting help for text messages.
Taking notes on BrailleNote	Cannot jot things down with paper/pencil.	Cannot read print even if she can write print.	Uses her BrailleNote as a notebook to store data.	*Independence.* Saves notes and phone numbers and helps plan the day.
Transferring data from BrailleNote to computer	Lacks ready access to transfer files.	Only other data access device is external floppy drive.	Uses floppy drive; reliable but slow and inefficient.	*Efficiency.* This inefficient method highlights unnecessary frustration.

point at which the user has implemented an alternative means of continuing the task. This alternative includes other methods of task completion, receiving outside help, or choosing to discontinue the task entirely. We are interested in the reasons for breakdowns and in the adaptive strategies, or workarounds, that are developed to carry out necessary tasks. "New design can be created and implemented only in the space that emerges in the recurrent structure of breakdown."[21] By focusing on the point at which a blind user detours from the designer's intended interaction, we begin to understand what motivates each workaround. We thus focus our data collection and analysis on the kinds of workarounds a nonsighted person adopts in carrying out everyday tasks and their implications for design.

Assistive technology. General guidelines exist for providing universal access to computing technology. One of the most influential is the W3C's *Web Accessibility Guidelines (WCAG) 1.0*,[23] which includes "Provide equivalent alternatives to auditory and visual content" and "Ensure user control of time-sensitive content changes." They sensitize designers to the fact that people interacting with their Web sites might not all have the same physical and cognitive abilities. Still, universal guidelines can easily obscure differences between people with different abilities, providing little guidance for designing for different interactional needs. For example, the guideline "Don't rely on color alone" from WCAG 1.0 is of little use in designing Web sites for people with total blindness.

Research focused on people with visual impairments has yielded a number of guidelines tailored more specifically to this population, such as "Non-speech sounds should be used to provide information and feedback about commands or events rather than verbal message"[16] and "[provide] multimodal feedback as a means of improving task performance, especially for individuals with visual impairments."[10] By employing a more ethnographically centered approach, we expand on previous studies, building on and complementing that research. Doing so, we hope to further understand how a blind user's experience and social context in addition to physical limitations affect the use of technology. Moreover, by more narrowly focusing on a computer user among the 0.03% of people in the U.S. with congenital blindness[5] (as opposed to, say, someone from among the 3.33% of the U.S. population with age-related macular degeneration[10,19]), we hoped to develop insights more specific to this smaller population. This is consistent with Newell and Gregor's concern[17] that "...except for a very limited range of products, 'design for all' is a very difficult, if not often impossible task" since "[p]roviding access to people with certain types of disability can make the product significantly more difficult to use by people without disabilities, and often impossible to use by people with a different type of disability."

Our interviews concerned not only usability but also aesthetics, affect, meaning, historical associations of use in context, and envisioning of future technologies.

We focused on someone with congenital blindness for two reasons: The first was personal, since the first author has a close friend with congenital blindness, and the project was inspired by informal discussion and interaction with this friend. The second was our belief that working with someone who had never had even residual sight would help highlight our taken-for-granted knowledge as researchers. Moggridge[15] captures this perspective when he wrote, "When we want to learn about people, it is important to include some who represent critical extreme values of the relevant characteristics, to avoid the trap of designing only for the average."[15] We undertook this research as sighted "outsiders," not as members of a blind community. In this regard, our perspective in relation to the research subject is much like that in a contextual inquiry,[3] where the design researcher seeks to understand the situated work practices in a particular setting through observation and discussion during the performance of the practices in situ.

Method

This case study of a congenitally blind college student, Sara (name changed to maintain confidentiality) took place in six sessions of approximately two hours each over a four-week time period in February and March 2006. The first author conducted all interviews, which were tape-recorded during each meeting as cotemporaneous notes were taken. These sessions were conducted in Sara's home, where she demonstrated tasks and shared her feelings about the artifacts used. Adapting Blythe et al.'s "Technology Biographies"[4] in these sessions, particularly "Technology Tours," "Personal History," and "Guided Speculation," we asked Sara to choose software and non-software artifacts to demonstrate and discuss.

She shared her BrailleNote, a chordal keyboard combined with refreshable Braille display and voice output, and demonstrated how she reads and writes using the device. She showed how she uses her cellphone to send and receive phone calls. She demonstrated her use of a Braille labeler to create embossed Braille tape she used to label the buttons on a microwave. She showed how she uses her Language Master (a voice-output electronic dictionary), including how she uses it as a thesaurus, to play games, and to look up words. She discussed her CD collection and demonstrated how she searches for and plays CDs. She demonstrated her screen-reading software (Job Access With Speech, or JAWS) by navigating a class-discussion Web site. She demonstrated how she uses plastic measuring cups for everyday cooking tasks and how she tells time using her tactile wristwatch.

We applied the Technology Tours, which involves asking how these items are used ("How would you go to a link in a Web page using JAWS?)" and observing her use of the object while concurrently listening to (and recording) her descriptions of her own actions. Using Personal History questions, we asked her to recall early memories about each object, as well as how she felt when she used it. Finally, using Guided Speculation, we asked her to describe any desire she had for each object or task in the future. Adapting Technology Biographies to our setting, we borrowed a protocol suited to a context-specific elicitation of technologies-in-use that are part of Sara's everyday world.

Sara demonstrated and discussed a variety of digital and non-digital artifacts she selected herself. We asked her to select them for two reasons: First, it allowed her to choose those she felt comfortable with and that were personally important to share in the context of the study. Second, allowing this breadth of artifacts extended the range of observations and topics discussed, contributing to the depth of the analysis of her interactions overall. In this sense, we can be confident that insights we draw across these digital and non-digital artifacts are representative of Sara's character and intentions.

We focus our data collection and analysis on the kinds of workarounds a nonsighted person adopts in carrying out everyday tasks and their implications for design.

Analysis

Throughout our note-taking and debriefings of interview sessions, we worked iteratively to capture our insights about limitations and workarounds, validating early conjectures with the subsequent data we collected. We shared our insights with Sara in subsequent interviews, soliciting her feedback and asking for additional clarification. We provide a brief example of this type of analysis for both a digital and a non-digital object in the remainder of this section.

We also summarize a sample of the data and insights from Sara's interactions in the table here. For most of the actions demonstrated, Sara had a workaround in situations where the default method failed her. For instance, with JAWS, Sara implemented a method of retracing her steps again until she was able to accomplish her task. Similarly, she navigated her extensive CD collection through a mix of spatially memorized locations and linear search. In each interaction, she negotiates efficient ways to accomplish her tasks. Other objects, such as tactile watch, cellphone, and labeler, reflect the importance of social context and independence on her choice of object and task or as a cause of frustration.

Tactile watch. Sara's tactile watch has Braille-like dots to mark the time on a clock face and a clear glass cover over it to protect the dots and watch hands. She easily flips open the lid to feel the time the hands point to. Interviews revealed her desire to avoid the kind of attention a talking digital watch might attract.

Sara: "I have a couple of talking watches, too. I just feel, I don't know like, I have a weird thing, I don't want to say that it's a bad thing or in any way put those things down, but I personally feel sort of embarrassed when I have to push a button and then people hear it and they're all like 'What is that?' and it just kind of draws attention to me, I feel like, in the wrong way—in a way I don't want attention drawn to me. So I just kind of try to avoid that as much as possible."

Unlike her talking watches, Sara found her tactile watch to be quiet, unobtrusive, and efficient at helping her tell the time. She also said that while the tactile watch was convenient and discreet enough for telling time, it lacked a built-in alarm function; instead, she relied on other electronic timekeeping devices for her morning alarm. She also described the delicate nature of the watch's physical makeup, sharing anecdotes of how easily the glass cover cracked or how frequently the batteries died and the inconvenience it caused. She also talked about preferring the aesthetic appeal and comfort of the tactile watch compared to her talking watches:

Interviewer: "Are there other preferences you have to your [tactile] watch, as opposed to the talking watches?"

Sara: "It's more comfortable... The other ones kind of look like big clunky sports watches. Sort of chunky. I just feel like it's more comfortable."

JAWS screen reader. Sara's JAWS screen reader works alongside her Windows operating system. She uses it to read aloud the text in the applications on her monitor by controlling an on-screen cursor through a series of hotkeys. Sara uses JAWS as a means to use her software applications: instant messaging, email, browsing the Internet, word processing, and backing up CDs.

Although JAWS increases her access to her computer, many interaction issues remain. For example, because JAWS is a text screen reader, it does not recognize pictures and graphics (ranging from chat emoticons to navigation tools on Web sites) and often gives vague feedback in describing where a graphic is placed in a document or Web page. One of the biggest challenges of using a screen reader is orientation and navigation. If Sara moves to another task or accidentally hits the wrong hot key, she might find herself in an unfamiliar virtual setting that requires her to suspend the current task, reorient herself, then resume where she left off. Her tenacity in the face of these obstacles is illustrated in the following transcript segment in which she is trying to navigate through frames on a course home page to get to a discussion board.

Braille watch with retractable glass cover and tactile numbers and hands.

PHOTOGRAPH BY NATASHA DUNN

Sara: "...I'm going to go back into the links list [JAWS speaks through the links in order: "communication, assignments, rules, contacts..."] no, silly, I wanted to go to discussion board. [tries a few more links, and the computer says them out loud] okay, it's not in the right place where I thought it was. Let me try that again, I'm sorry. [starts through the links list again] Okay, discussion board, I'm tabbing through this time and not going through the links [the computer talks] I'm on there, c'mon go back to the discussion board [silence, then the computer speaks] come on... Go to the discussion board. Now. [the computer speaks again] okay, let's try this again. I'm going to right-click it. I press the right mouse button, which is this one... Okay, press Enter and see if I go anywhere. Why is it misbehaving?"

At this point, having tried all that she could think to do, Sara is frustrated and anxious to move on. It is only on starting over by reentering the URL of the Web page and carefully stepping through each action that she finally is able to find the discussion board.

Sara: "Okay, now it's taking me back to the home page. So let's try this again. Okay, I'm going back into the links list. Pressing Enter. Come on... [quiet for some time; the computer appears to be opening the link and suddenly speaks again] Oh, here we go. [the links list appears on the screen] Okay, now let's go to the discussion board. 'D' for discussion board. [the computer goes through the details for the page, a heading, the number of links, and more]."

Sara employs two specific, strategic workarounds here. First, she tries all the options available to her. When none lead to the expected outcome, she aborts the original operation and begins again. Both tactics are brute-force, when-all-else-fails solutions that are time-consuming and sometimes frustrating but that are most likely to yield desired results. As multiple programs are always running on the computer, just making a diagnostic check of where things are "located" can be time-consuming and difficult. This sometimes poses limitations on the usability of JAWS; Sara's workaround here is to repeatedly try different operations until her intentions are fulfilled. From this experimentation and practice, she is able to learn pragmatically what works and what doesn't.

Though the quotes indicate the considerable usability problems Sara encounters in using JAWS, they also affect issues of socially situated meaning. As a student, she relies on the computer and Internet for social connections and course-related communication. Raising the cost of performing relatively simple operations that are error-prone and resistant to efficient workarounds affects Sara's ability to access the information required to be a full participant in courses and engage in social interaction online.

Insights

Having understood limitations and workarounds in isolation, we identified recurrent themes common across objects and tasks. Sara's actions and associations with objects and tasks were guided by both the usability of the object and the meaning she accorded to the task. A stronger personal preference or significant item or task often motivated her to overcome physical obstacles at almost any cost. In the table, the "Workaround" column lists the alternatives Sara employed to get around limitations. Additionally, we added personal assessments Sara might have developed on each workaround, showing her distaste for very inconvenient workarounds, such as getting a friend to help. Often these unsatisfactory workarounds were avoided, generally indicating the task was also avoided. When Sara was required to type in the letters shown in a picture to gain access to a Web site, her frustration with the workarounds (call tech support or get sighted help) led her to drop any Web site that required such actions. Items or actions that held little personal significance were easier to pass up if the physical interaction was too time-consuming or made her uncomfortable.

Our focus on usability and socially situated meaning generated several insights based on Sara's workarounds into what motivates her use of an artifact. The following sections provide a general classification of the issues Sara faced when interacting with artifacts and technology. Specific, personal preferences included her motivation to seamlessly engage with her environment, a world often contextualized for sighted people. Facets of design included those areas of interaction that caused her frustration, such as lack of control, or that created or eliminated barriers to content, such as tactile feedback.

Socialization within a predominantly sighted community. Several of Sara's decisions reflect her desire to be included in her community of sighted friends and family and to include others in her life as smoothly as possible. Some choices she makes include using a tactile watch and prominently displaying a bulletin board of print-labeled photographs on her wall. Asked why she had these labeled photographs, she said it was as a conversation piece for when sighted friends visit. She also said "I'm the only blind person on campus and I don't know, I just try to integrate myself into the world and in that sense, you know, as much as possible." It is important to consider design ideas supporting cohesive socialization with the people within Sara's social sphere. Showing off her BrailleNote, she said she prefers reading Braille, as opposed to listening to talking software, because it is quiet.

She also said that carrying around her awkwardly shaped labeler makes her feel self-conscious and expressed frustration when she is not acknowledged in casual social situations due to her blindness. A concrete design modification she suggested is to allow a Braille labeler to make print labels. A dual print labeler would allow her to create labels so she could better share mixed CDs she makes for friends.

Braille labels help identify objects and content.

Independence. Sara is independent and tackles issues from multiple sides until she reaches a solution. Object design should support her ability to be independent and not require sighted help. Sara's independence was highlighted when she talked about taking a cab when needed, rather than relying on friends and relatives for transportation. She relishes the freedom her cellphone gives her, providing easy access to others only if in need.

Control. For Sara to be able to maintain her independence, she must be able to control significant factors that ultimately help her accomplish her tasks. Design should grant the user full control of as many functions as possible and allow switching between interaction modes in different contexts. Evidence of Sara's desire to be in control came from her tendency to stick with tasks and objects she finds comfortable, avoiding things she can't do, such as going to Web sites with special accessibility pages. In working with JAWS software, she showed tenacity in trying all possible options before asking for sighted help.

Efficiency. Compensating for sight is often time-consuming; for example, if Sara does not remember where she has placed a CD, she must carry out a linear search—pulling out a single disc case from its position on her CD shelf, reading the Braille label on the case, replacing the CD, and moving on to the next. For enhanced usability, efficiency is an important factor to consider in an object's design. Sara's accurate memory and learned procedures help her use certain items quickly. This efficiency allows her to focus on the enjoyment of certain items and tasks, such as using her CD player, rather than on the mechanics of carrying out the tasks. Conversely, inefficiency increased her frustration and time to completion, such as when she had to reorient herself while using JAWS.

Portability. Sara's strong ties to her cellphone can be attributed, in part, to its small, easy-to-carry size. In contrast, she expressed annoyance toward objects that were not as easily portable, such as her large and awkwardly shaped Braille labeler. Object portability increases efficiency, supports independence, and eases integration within Sara's social world.

PHOTOGRAPH BY MOIRA CLUNIE

Distinguishability of similars. Usability increases when Sara is able to distinguish among similar item features. Where a sighted user distinguishes similar features, such as among the buttons on a CD player or CD cases in a collection, due to written labels on the items or by seeing each item's position in a larger spatial context, Sara had much more difficulty. She labeled items in Braille that were otherwise difficult to distinguish and used her hands, fingers, feet, nose, and ears to see what her eyes could not. Each opportunity for sensing and distinguishing can be exploited by technology designers. Design that aids identification and distinguishability of otherwise similar features, such as CD cases with preprinted Braille identifiers and cellphone keys with textured surfaces, enhances ease of use, flexibility, and efficiency. Sara showed how, with deft fingers, she is able to distinguish different-size measuring cups held together by a common ring. She complained that some cellphones lacked keys she could identify by touch.

Brute-force backup. One fallback problem-solving technique is to exhaustively try all possibilities. When Sara became disoriented while demonstrating her Language Master and JAWS software, she tried all possible options as much as possible. Due to her self-described disorganization, her linear search method for CDs is the most effective tactic, but also very time-consuming.

Flexibility and interoperability. Sara took notes and read books on her BrailleNote, but her model lacks external storage, except for a floppy drive, and does not include access to the Internet. By considering how people will use particular devices when carrying out larger task landscapes,[14] such as that Sara might not only want to take class notes but share these notes with friends using the Internet, designers increase opportunities for use. Despite the many uses of her BrailleNote, Sara wanted a laptop computer due to its flexibility for Internet access, communication, and storage.

Simply replacing one interaction mode, such as the display of text on a screen with a functionally equivalent mode, as in speaking the text aloud, is not necessarily equivalent from the point of view of user experience.

Conclusion

We draw two main conclusions from the study. The first is methodological. We used an ethnographically inspired investigation of a single, nonsighted individual interactions with a variety of artifacts in her own home. Our concern with not only usability but socially situated meaning contrasts with experimental designs focused on measurement and control and with lab-based usability studies. Although we can state our conclusions as principles associated with this individual's preferences and beliefs, such statements are animated in the ways in which we observed them. We are unsurprised that, for instance, Sara's sense of self is intimately tied to her relationships in her social network. How could it not be? We are, however, surprised in the specific ways in which this was embedded in her choices about artifacts and interactions. Such surprise—in her wall of textually labeled photographs as conversation pieces for her sighted friends, her preference for a tactile watch because of how it looks and feels, and the use, when she was a child, of her talking dictionary as an ice-breaker with friends—undoubtedly reflects our situatedness (as researchers and as people) within our own social worlds, the taken-for-granted assumptions we carry with us as sighted people.

To what extent are we able to generalize these results to other contexts and other people? Though single-person case studies are rare in human-computer interaction, they have a long history in the social and behavioral sciences. For example, studies with single participants that have been influential were undertaken by Freud in psychoanalysis,[7] Harper in sociology,[9] and Luria in brain/mind studies.[12] One goal of case-study research is to develop theoretical propositions that can be used to guide subsequent research studies and design efforts. However, it is important to understand the difference between these analytic generalizations and the statistical generalizations common in experimental study designs. Emphasizing this distinction, Yin[24] writes, "'How can you generalize from a single case?'... The short answer is that case studies, like experiments, are generalizable to theoretical propositions and not to populations or universes. In this sense, the case study, like the experiment, does not represent a 'sample,' and in doing a case study, your goal will be to expand and generalize theories (analytic generalizations) and not to enumerate frequencies (statistical generalization)." He further points out, "Scientific facts are rarely based on single experiments; they are usually based on a multiple set of experiments that have replicated the same phenomenon under different conditions."

We do not claim that our results generalize to all people in any particular group, not even provisionally. Neither do we claim that Sara is typical of any group, such as people who are blind or people with physical disabilities. However, there is nothing so particular about Sara that precludes these results from applying to others who might be similarly situated within their physical and social worlds.

Our focus on a single individual across a range of tasks and artifacts allowed us to seek coherence in the themes and patterns that spanned many aspects of her life, something we might have missed had we instead looked at a small range of tasks and artifacts across several individuals. Also, the range of tasks, including those involving both computational and noncomputational artifacts, increases our confidence that we captured the key issues characterizing Sara's interaction with technology. Enabling more access to technology may in fact require that we look at increasingly specific populations so as to tailor technologies more closely to people's needs. However, our case study, with its rich data across a variety of interactions, provides a set of hypotheses that can be used comparatively in other studies.

Our second conclusion reiterates that it is the combination of functionality and socially situated meaning that determines who will use technology and how it will be used. Simply replacing one interaction mode, such as the display of text on a screen with a functionally equivalent mode as in speaking the text aloud, is not necessarily equivalent from the point of view of user experience. This is because functional equivalence might not account for the meaning of the mode of interaction for particular users in specific contexts. Efforts to provide multimodal support for people with perceptual and/or motor disabilities[10,22] are encouraging, not simply because of the increased physico-cognitive support they provide. Rather, multimodal support offers the possibility of using different modalities on different tasks and in different contexts, but only if the designer allows this degree of user control of interaction mode.

Paradoxically, increasing and universalizing access to technology might require attending to the specific and situated meanings of technology use by particular populations in particular settings. Because technologies invisibly embed taken-for-granted assumptions concerning trade-offs in functionality, usability, and situated meaning, developing an understanding of these trade-offs for particular people and populations can improve technology access for increasing numbers of people.

Acknowledgments

We thank Sally Fincher for initial project brainstorming, key bibliographic references, and feedback on drafts; Genevieve Bell for helping overcome challenges in subject recruitment and for suggestions on carrying out ethnographic fieldwork; Lisa Tice for utilizing her professional network to help with recruitment; Mark Blythe for providing information on his protocols; Skip Walter and Eli Blevis for feedback and encouragement as we were completing our fieldwork; Donald Chinn and Carol Hert for support throughout; and Youn-kyung Lim for her many suggestions for improving the reporting of these results. And thanks most especially to Sara for her benevolence, sincerity, and willingness to share her life and to Laurie Rubin for her inspiration, motivation, and support.

This is a revised version of an earlier paper presented at the Ninth Interactional ACM SIGACCESS Conferences on Computers and Accessibility (Tempe, AZ, Oct. 15–17, 2007); http://www.sigaccess.org/assets07/ and appears at http://portal.acm.org/citation.cfm?doid=1296843.1296873.

References

1. Basso, K. *Wisdom Sits in Places.* University of New Mexico Press, Albuquerque, NM, 1996.
2. Bell, G., Blythe, M., and Sengers, P. Making by making strange: Defamiliarization and the design of domestic technologies. *ACM Transactions on Computer-Human Interaction 12*, 2 (June 2005), 149–173.
3. Beyer, H. and Holtzblatt, K. *Contextual Design.* Morgan Kaufmann Publishers, San Francisco, CA, 1998.
4. Blythe, M., Monk, A., and Park, J. Technology biographies: Field study techniques for home use product development. In *CHI02 Extended Abstracts on Human Factors in Computing Systems* (Minneapolis, MN, Apr. 20–25). ACM, New York, 2002, 658–659.
5. Bouvrie, J. and Sinha, P. Visual object concept discovery: Observations in congenitally blind children and a computational approach. *Neurocomputing 70*, 13–15 (Aug. 2007), 2218–2233.
6. Csikszentmihalyi, M. and Rochberg-Halton, E. *The Meaning of Things: Domestic Symbols and the Self.* Cambridge University Press, Cambridge, U.K., 1981.
7. Freud, S. *Dora: An Analysis of a Case of Hysteria.* Collier Books, New York, 1963.
8. Geertz, C. *The Interpretation of Cultures.* Basic Books, New York, 1973.
9. Harper, D. *Working Knowledge: Skill and Community in a Small Shop.* University of Chicago Press, Chicago, 1987.
10. Jacko, J., Moloney, K., Kongnakorn, T., Barnard, L., Edwards, P., Leonard, V., and Sainfort, F. Multimodal feedback as a solution to ocular disease-based user performance decrements in the absence of functional visual loss. *International Journal of Human-Computer Interaction 18*, 2 (2005), 183–218.
11. Kurniawan, S., Sutcliffe, A., Blenkhorn, P., and Shin, J. Investigating the usability of a screen reader and mental models of blind users in the Windows environment. *International Journal of Rehabilitation Research 26*, 2 (June 2003), 145–147.
12. Luria, A. *The Man with a Shattered World: The History of a Brain Wound.* Harvard University Press, Cambridge, MA, 1987.
13. Magnusson, C., Rassmus-Grohn, K., Sjostrom, C., and Danielsson, H. Navigation and recognition in complex haptic virtual environments: Reports from an extensive study with blind users. In *Proceedings of Eurohaptics 2002* (Edinburgh, U.K., 2002).
14. Mirel, B. *Interaction Design for Complex Problem Solving.* Morgan Kaufmann Publishers, San Francisco, CA, 2004.
15. Moggridge, B. *Designing Interactions.* MIT Press, Cambridge, MA, 2006.
16. Morley, S., Petrie, H., O'Neill, A., and McNally, P. Auditory navigation in hyperspace: Design and evaluation of a non-visual hypermedia system for blind users. *Behaviour & Information Technology 18*, 1 (Jan. 1999), 18–26.
17. Newell, A.F. and Gregor, P. User-sensitive inclusive design—In search of a new paradigm. In *Proceedings on the 2000 Conference on Universal Usability* (Arlington, VA, Nov. 16–17). ACM Press, New York, 2000, 39–44.
18. Spradley, J. *Participant Observation.* Thomson Learning, 1980.
19. U.S. Census Bureau. *Population Estimates 2000 to 2006*; http://www.census.gov/popest/states/NSTannest.html.
20. Wall, S. and Brewster, S. Providing external memory aids in haptic visualisations for blind computer users. In *Proceedings of the Fifth International Conference on Disability, Virtual Reality, and Associated Technologies* (Oxford, U.K., Sept. 2004).
21. Winograd, T. and Flores, F. *Understanding Computers and Cognition: A New Foundation for Design.* Ablex Publishing Corporation, Norwood, NJ, 1986.
22. Wood, J., Magennis, M., Arias, E., Gutierrez, T., Graupp, H., and Bergamasco, M. The design and evaluation of a computer game for the blind in the GRAB haptic audio virtual environment. In *Proceedings of Eurohaptics 2003* (Dublin, U.K., July 2003).
23. World Wide Web Consortium. *Web Content Accessiblity Guidelines 1.0*, 1999; http://www.w3.org/TR/WCAG10/.
24. Yin, R. *Case Study Research, Design and Methods. Applied Social Research Methods Series, Vol. 5, Third Edition.* Sage Publications, Thousand Oaks, CA, 2003.

Kristen Shinohara (kshino@u.washington.edu) is a Ph.D. student in the Information School of the University of Washington, Seattle, WA.

Josh Tenenberg (jtenenbg@u.washington.edu) is Professor in Computing and Software Systems in the Institute of Technology of the University of Washington, Tacoma, Tacoma, WA.

DOI:10.1145/1536616.1536635

The humanitarian focus of socially useful projects promises to motivate community-minded undergrads in and out of CS.

BY RALPH MORELLI, ALLEN TUCKER, NORMAN DANNER, TRISHAN R. DE LANEROLLE, HEIDI J.C. ELLIS, OZGUR IZMIRLI, DANNY KRIZANC, AND GARY PARKER

Revitalizing Computing Education Through Free and Open Source Software for Humanity

WHAT IF UNDERGRADUATE students viewed computer science as, in part, a discipline that designed and built free software to help one's friends and neighbors in need? Would that bring more of them in the front door of academic computing departments? What sort of curricular and pedagogical changes would be required to support such opportunities for these students? Would these changes help revitalize computing curricula and enrollments throughout the U.S.?

The Humanitarian Free and Open Source Software (HFOSS) Project is addressing these questions. The goal is to help revitalize U.S. undergraduate computing education by engaging students in developing FOSS that benefits humanity. What started as an independent study by two undergraduates in 2006, the Project today includes students from a number of U.S. colleges and universities engaged in a range of FOSS development projects, both global and local. Here, we provide an overview of the Project, along with some of the lessons learned and the challenges that remain. Our experience over the past three-and-a-half years suggests that engaging students in building FOSS that serves society is a positive step toward strengthening undergraduate computing education.

The Project has been supported since September 2007 by a National Science Foundation CPATH[a] grant, aiming in part to build a collaborative community of individuals from multiple educational institutions, computing and IT organizations, and nonprofit social-service agencies to support undergraduates in the development of socially useful FOSS. In general, the Project aims to answer whether getting students involved in humanitarian FOSS indeed also helps revitalize undergraduate computing education.

Inspiration came from the Sahana project, a FOSS disaster-management system developed by a group of Sri Lankan volunteers in the aftermath of the December 2004 Asian tsunami. The Project began working with Sahana in January 2006 after the author Trishan de Lanerolle learned about it during a visit to Colombo, Sri Lanka (see the sidebar "Five HFOSS Software Projects"). That spring, two Trinity College students installed Sahana on an Apache server and began exploring its LAMP architecture (Linux/Apache/MySQL/PHP) as part of their independent-study course. They worked with the code and seemed to enjoy the opportunity and challenge of being engaged in a real-world software project (as opposed to a class exercise). That summer, with support for a research student and in collaboration with community-minded volunteers from Accenture Corporation (http://accenture.com), it developed a volunteer-management module that was eventually accepted into Sahana's code base. Thus began an ongoing collaboration with the Sahana community.

The initial experience with Sahana dovetailed with two ideas outlined by former ACM president David Patterson in his "President's Letter" columns in *Communications*. In "Rescuing Our Families, Our Neighbors, and Ourselves" (November 2005), he suggested it might be the civic duty of computing professionals to be more involved in helping their communities recover from natural disasters while simultaneously helping the profession.[5] In "Computer Science Education in the 21st Century" (March 2006), he explored the disconnect between how programming is taught in the classroom and how cutting-edge software is written in industry, urging educators to involve themselves in the open source movement.[6]

The call to build open source software to help our neighbors resonated with the Sahana experience, suggesting that a project organized around this theme might yield beneficial outcomes for undergraduate computing:

- Give computing students experience with the open source development process in a real-world practitioner environment;
- Let them see firsthand the importance of software-engineering principles;
- Enable them to use their programming and problem-solving skills to contribute to the expanding volunteerism movement that characterizes many of today's college campuses;
- Make it possible for them to gain firsthand contact with IT professionals in the computing industry;
- Enable computing faculty to experiment with a variety of approaches for incorporating FOSS into the curriculum; and
- Invite all participants—faculty, students, IT professionals, and the humanitarian community—to join in a mutually beneficial educational and social enterprise.

Problems that beset undergraduate computing education in the U.S. include sagging enrollment, out-of-date curricula, changing demographics, and rapidly evolving technologies. While they are most closely associated with the academic computing discipline, they are also associated with a number of myths and misconceptions that extend well beyond the academy to society in general: computer science is nothing but coding; computing students are geeks; programming is an isolating activity; and computing jobs are being outsourced to Asia and Eastern Europe.

These problems and myths cannot be addressed within the academy alone. Rather, what's needed is a sustained effort involving a broad coalition of computing educators and industry professionals. Only such an effort can change false perceptions about computing in the larger society. The effort also requires substantial support from the computing industry, which stands to benefit from a revitalized computing curriculum. It also may require the kind of infrastructure and publicity one finds in other communities (such as Teach for America and Habitat for Humanity) that attempt to mobilize students and others to take on real-world projects for the social good.

Serve Society

While FOSS applications run the gamut of computer software, HFOSS, as we define it, is software that serves society in some direct way. This deliberately broad definition is meant to be inclusive of a wide range of socially beneficial projects and activities.[b] To date, the HFOSS Project has not had to face the question of where to draw the line between humanitarian and non-humanitarian FOSS. As a practical measure we use the guideline that any software artifact the Project creates must intrinsically benefit a nonprofit organization pursuing some kind of public-service mission.

As described by Chopra and Dexter[2] the free-software movement has roots going back 60 years to the beginning of the computer age when sharing programming ideas and code was the norm. The modern free-software movement began in 1983 when Richard Stallman defined "free software" as the freedom to use, study, copy, change, and redistribute software "so that the whole community benefits" (http://www.gnu.org/philosophy/free-sw.html).

Following the spectacular success of the GNU/Linux project (http://www.gnu.org/), the free-software movement has grown in scope and importance. An April 2008 study by the Standish Group (http://www.standishgroup.com/) estimated that open source software costs the software industry $60 billion in potential annual revenue.[9] SourceForge (http://source-

a CPATH is a National Science Foundation program within the Directorate for Computer & Information Science & Engineering, formally known as CISE Pathways to Revitalize Undergraduate Computing Education (http://www.nsf.gov/cise/funding/cpath_faq.jsp).

b See also a similar definition in http://en.wikipedia.org/wiki/Humanitarian-FOSS.

2008 HFOSS summer-internship program students and faculty, Trinity College (http://2008.hfoss.org).

PHOTOGRAPH BY ELI FOX-EPSTEIN

forge.net), the primary open-source hosting site, lists more than 180,000 projects and 1.7 million registered users worldwide. Many top software and Internet–related companies, including Dell, Google, Hewlett-Packard, IBM, Intel, and Microsoft, support the FOSS model in one way or another. According to an August 2008 Linux.com article, students are beginning to join open source projects as a way to gain relevant work experience needed for many entry-level computing positions (http://www.linux.com/archive/feature/143415).

The free-software movement is characterized by the way it distributes its products. The GNU General Public License (GPL) was the first of many free-software licenses stipulating how the software can be freely used and shared. As Stallman wrote, software freedom, in this sense, is "a matter of liberty, not price"; it is free as in free speech and not (necessarily) as in free beer. The free-software philosophy is supported and promoted by the Free Software Foundation (http://www.fsf.org).

The free-software movement is also characterized by an open development process, a highly distributed, nonhierarchical, peer-based activity. The FOSS approach stands in sharp contrast to the top-down, hierarchical, legacy-based model of traditional commercial software development. This distinction is often exemplified by the difference between how Linux and Microsoft Windows were developed. FOSS programmers collaborate in loosely organized communities, freely working on the projects and problems that are of most interest to them. The FOSS development process is also closely tied to the user community and marked by frequent releases closely monitored and tested by end users. To use a metaphor coined by Eric Raymond, author of *The Cathedral* and *The Bazaar*,[7] the free software development process resembles a "babbling bazaar," unlike the "cathedral" model historically employed in commercial software development.[7]

The free-software movement split into two competing philosophies in 1998 when a group led by Raymond and Bruce Perens co-founded the Open Source Initiative (OSI) to make free software more commercially attractive (http://www.opensource.org). OSI has since become the steward of the open source definition and serves (together with the FSF[8]) as a standards body for vetting and approving open source licenses, of which there are dozens (http://www.opensource.org/licenses/alphabetical). As reflected in its name, the HFOSS Project accepts the principles and practicalities of the FOSS movement as characterized by both FSF and OSI.

Since spring 2006 the HFOSS Project has engaged students from Bowdoin College, Connecticut College, Trinity College, Wesleyan University, the University of Connecticut, and the University of Hartford in a number of software-development projects serving the community. Its main software-development activities take place during its annual 10-week summer internship program, now in its third year (see the figure here). But students also work on HFOSS projects in courses, independent studies, and thesis projects (outlined in the sidebar).

Given its primary goal of contributing to the revitalization of undergraduate computing education, the HFOSS Project has six specific objectives that, if met, would represent significant progress toward its overall community building and revitalization goal:

- Introduce new concepts and methodologies;
- Attract a new demographic;
- Debunk the computing-is-coding myth;
- Unite town and gown;
- Contribute to society; and
- Create a portable and sustainable model.

Concepts and Methodologies

As a concept, HFOSS is clearly attractive to university computer science students and may help attract new students to computing. This is reflected not only in the interest that has been generated in the summer HFOSS Institutes, where typically two to three times more students apply than can be accommodated but also in the feedback we receive from students in HFOSS software-engineering and software-development courses throughout the curriculum.

Five HFOSS Software Projects

The first three projects are international in scope and involve students in global communities and ongoing software development. The other two projects engage students in local nonprofit organizations to develop custom software that helps the organizations directly. Participation in all five depends on Internet-based communication, collaboration, and software-development technologies. In addition to list servers for shared email messages, students use wiki pages and version-control repositories to share documentation and code with one another and with their mentors. Development teams in each project hold regular virtual meetings through videoconferencing and Internet relay chat.

Sahana. Sahana (Sinhalese for relief) is a FOSS disaster-management system built initially by Sri Lankan volunteers in the aftermath of the 2004 Asian tsunami (http://www.sahana.lk). It addresses the common coordination problems that arise during disaster recovery—finding missing people, managing aid and volunteers, and otherwise assisting the effort. It has also been deployed in numerous disasters around the world, most recently in the 2008 Burma cyclone and 2008 earthquake in China. From its beginnings in Colombo, Sri Lanka, it has grown into a worldwide community of individuals and organizations that support ongoing development, receiving the 2006 Social Benefit Award from the Free Software Foundation (http://www.fsf.org/social-benefit-award-2006).

Beginning in January 2006, our involvement with Sahana focuses on development and support of the volunteer-management (VM) module incorporated into the Sahana code base in December 2006.[3] A first prototype of the module, which supports registration and management of relief volunteers, was field-tested with Sahana at the June 2006 Strong Angel III Disaster Response Demonstration in San Diego (http://www.strongangel3.net). One student said, "This isn't a typical computer science project. How many students get to publish software that can potentially affect millions of people's lives?" (http://www.trincoll.edu/AboutTrinity/News_Events/trinity_news/061013_sahana.htm).

Over the past three years HFOSS has become both an integral contributor to and a beneficiary of the Sahana community. The author Trishan de Lanerolle now serves on Sahana's management committee, and two HFOSS alumni have been granted "committer" status, giving them direct access to the Sahana repository.

On the educational side, Sahana has been used as a teaching platform in numerous courses, independent studies, and summer-internship projects.[c] Students in the 2007 HFOSS Summer Institute performed a complete refactoring of the VM module based on a model-view controller design.[d] In spring 2008 Sahana was used as the software platform for an introductory course involving 13 Trinity and Wesleyan students.[4] And in summer 2008 a team of four undergraduates developed a credential-verification module under the direction of Frank Fiedrich of George Washington University's Institute for Crisis, Disaster, and Risk Management (http://www.gwu.edu/~icdrm).

Figure 1. Field testing the Sahana volunteer-management module at the Strong Angel III disaster response exercise, San Diego, CA (http://www.strongangel3.net).

In May 2008, in an engagement that illustrates how the HFOSS community makes a positive contribution, a team of six students and faculty worked closely over 10 days with a Sahana team in Sri Lanka and an IBM team in China to support deployment of Sahana in Chengdu following the devastating earthquake there (see Figure 1). One China team member later said, "It was really an emotional moment of truth when we saw the happy tears as people were reunited with their families. Eventually, we can say with pride that what we have done is worth remembering for our whole life. We helped people in the disaster area with our technology" (see Figure 2) (http://blog.hfoss.org?cat=29).

Finally, in keeping with the sharing nature of FOSS culture and licensing, the VM module has found application beyond the Sahana system and disaster-recovery domain. For example, a modified version of the original VM module is now incorporated into a coastal-flood emergency-preparedness system for the New York City Office of Emergency Management. The system is designed to manage the potential evacuation of 1.2 million people from low-lying areas and shelter 600,000 evacuees in temporary shelters. Similarly, the Office of Emergency Management in Galveston City, TX, is looking at Sahana and the VM module for its own disaster-preparedness purposes. Using Sahana as the basis for other disaster-preparedness systems could provide a way for students in many schools around the U.S. to involve themselves in HFOSS development projects.

Open Medical Record System (OpenMRS). This FOSS electronic medical record system assists health professionals in the treatment of AIDS, malaria, and tuberculosis in the developing world, particularly in Africa (http://www.openmrs.org). The project began in 2004 as a joint venture of the Regenstrief Institute (http://www.regenstrief.org) and Partners In Health (http://www.pih.org), aiming to provide health-care professionals the information-management tools they need to combat these diseases and provide quality care. OpenMRS has since been deployed in Kenya, Lesotho, Mozambique, Rwanda, South Africa, Tanzania, Uganda, and Zimbabwe.

Like Sahana, the OpenMRS community is a worldwide network of individuals and organizations contributing to the development of the software. It makes extensive use of Java-based development technologies, including Java server pages and servlets, the Spring application framework, and other advanced FOSS tools. Unlike Sahana's relatively simple PHP/MySQL platform, OpenMRS is substantially more complex and challenging. Nevertheless, HFOSS students have made several important contributions to the OpenMRS project.

During summer 2007, they developed a module to enable the system to be used with a touchscreen monitor, an effort that continued as a senior thesis project and resulted in a generic touchscreen application, AutoTouch, providing an API to add a touchscreen interface to any Web-based application (http://sourceforge.net/projects/autotouch). During the summer 2008 HFOSS Institute, students created an OpenMRS module for uploading and editing patient medical images and

c These and other activities involving Connecticut College, Trinity College, and Wesleyan University are funded by a Mellon Foundation grant for videoconferencing facilities.

d The 2007 HFOSS Summer Institute was funded by a grant from the Aidmatrix Foundation (http://www.aidmatrix.org).

defined and implemented a new systemwide data structure that allows physicians to input numeric observations as ranges (1–5), inequalities (<100), ratios (2:5), and qualitative values (too few to count).

During spring 2009, a software-development course based on OpenMRS was offered (via videoconference) to students at Connecticut College, Trinity College, and Wesleyan University. Focusing on how software-engineering techniques play out in an FOSS setting, it required students to put theory into practice by contributing to OpenMRS.

Innovative Support To Emergencies, Diseases and Disasters (InSTEDD). This lab is devoted to developing software for early disease detection and disaster response (http://www.instedd.org). Founded in 2006 by Larry Brilliant of the Google Foundation, it is funded in part by Google and the Rockefeller Foundation (http://www.rockfound.org) and aims to integrate, tag, classify, and visualize data from various sources (such as news, weather reports, sensor data, and field reports) with the goal of detecting and managing disease outbreaks. Like Sahana and OpenMRS, InSTEDD is an international effort.

Two students in the summer 2008 HFOSS Institute collaborated with researchers in Seattle and Buenos Aires to develop and test data mining algorithms for the Evolve project. After studying support-vector machines and Bayesian networks and mastering software tools (such as Eclipse and LIBSVM, http://www.csie.ntu.edu.tw/~cjlin/libsvm/), they developed the Alpaca Light Parsing and Classifying Application (ALPACA), a parsing and classification tool for categorizing documents into user-provided classes (http://2008.hfoss.org/ALPACA). ALPACA allows Evolve developers and others to test classification technologies across a number of data sets. Two other students are following up on this work as part of the 2009 HFOSS Summer Institute.

Ronald McDonald House Homebase. This project involves a Web-based volunteer-management-and-scheduling system used at the Ronald McDonald House in Portland, ME (http://www.rmhportland.org). Developed in spring 2008, it addresses the need for software to replace the Portland Ronald McDonald House's error-prone, time-consuming manual rolodex and calendar-scheduling process. The development team included four Bowdoin College students, one professor, Bowdoin IT staff, and several Ronald McDonald House employees and volunteers who would eventually use the system. The development took place almost entirely within a software-development course (http://hfoss.bowdoin.edu) using a distributed-development process and the same FOSS tools used in the global projects.[10]

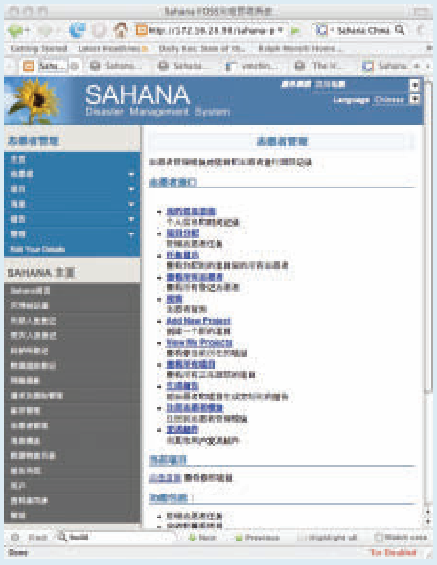


Figure 2. Screenshot of the Chinese-language version of the volunteer-management module (http://blog.hfoss.org/?p=28).

The four students earned independent-study credit in computer science, as well as a valuable learning experience. The Portland Ronald McDonald House gained a valuable piece of software that arguably would never have been developed outside the FOSS framework. The software is published on Sourceforge (http://www.sf.net/projects/rmh-homebase) under a GNU GPL and is available to other volunteer organizations.

One difference between this project and the three international projects is the software was designed and built from scratch, though it followed careful study of the Sahana system. Also, unlike the international projects, it involved close interaction with end users throughout the development process. It also provides a potential basis for groups of students and faculty at other colleges and universities to join in by, say, customizing and adapting the system for other Ronald McDonald Houses or other local nonprofit organizations.

AppTrac. Literacy Volunteers of Greater Hartford provides literacy training to adults, mostly through specialized software applications in its Hartford, CT, computer lab. The staff manually tracks student logins, the applications the students use, and other information it then painstakingly compiles into reports to the organization's board and funding agencies.

In spring 2008, students from the University of Hartford developed requirements and built a prototype application-tracking system (AppTrac) as part of an upper-level software-engineering course. During the 2008 HFOSS Summer Institute a four-student team from three colleges—Connecticut College, Trinity College, and the University of Hartford—developed the prototype into a kiosk-based system (http://sourceforge.net/projects/apptrac/). In fall 2008, working through virtual meetings, code-sharing repositories, and wikis, the same students modified the system for eventual release as a generic kiosk system for similar applications.

Unfortunately, due to the loss of its technical staff position, the Literacy Volunteers of Greater Hartford ultimately decided not to deploy AppTrac in its lab. However, the software is being modified by students in the 2009 HFOSS Summer Institute (http://2009.hfoss.org) for deployment in the Hartford Public Library's computer lab, another example of how software sharing benefits both the community and the educational process.

To explore this concept further, in spring 2008 a "general education" course called "Open Source Software for Humanity," was taught (via videoconference) at Trinity College and Wesleyan University.[4] Its "hook" was getting students to reflect on their own experience with FOSS products (such as Wikipedia and the Firefox browser). Not surprisingly, the students were receptive to the ideals of sharing, community, and the public good. They were also enthusiastic about discussing their experience with Wikipedia, blogging, open source politics, and other aspects of the free and open culture they had grown up with. As suggested by Benkler and Nissenbaum,[1] they see the distributed FOSS model as an alternative means of producing culturally useful goods (Wikipedia) and services (SETI@home). Similarly, students generally see elements of the FOSS ethic in their own experience with file sharing. They recognize that this is a time of change in public thinking about intellectual property and the common good.

But despite their everyday use and enjoyment of FOSS products and their widespread acceptance of the freedom and openness characterizing the FOSS model, few students recognize the connections between the FOSS movement and the overall computing discipline. As one said, "Wow, I really got to look at how computer science can relate to humanitarian efforts. I now really understand [FOSS] and know why it came about."

As a methodology, the FOSS development model represents a revolutionary break from traditional software development.[7] However, despite its commercial success, relatively little effort has gone toward incorporating the FOSS development model into undergraduate computing curricula. Our effort to see how others have incorporated FOSS into their curricula revealed only a handful of reports (reviewed by Ellis et al.[3]). Our experiments with introductory and advanced courses, independent studies, and summer internships have shown that FOSS software and tools, including Apache, PHP, MySQL, Eclipse, PhpMyAdmin, and SVN, are quite accessible to today's undergraduates.

Students are also eager to engage the HFOSS methodology, which differs from the traditional mode of undergraduate instruction. Working with mentors and in teams on real-world development projects is a motivator for students, despite the extra challenge it means for instructors. Similarly, working with local clients and international development communities is another motivator. For example, students get to see directly that writing good documentation is as important as writing good code. The quality of their work improves as they recognize their increased level of public accountability. This message is constantly reinforced by mentors, peers, and clients.

Depending on the specific course or project, students come with different levels of expertise, ranging from no prior programming experience for an introductory course to having nearly completed the major requirements for upper-level and software-engineering courses. Engaging students through HFOSS must be done with sensitivity to their backgrounds and interests. But the projects themselves are rich and varied enough to accept contributions from students with different backgrounds. For example, students with no programming experience are still able to make significant contributions in requirements-gathering and documentation-writing.[4]

We've found the sudents are comfortable working in virtual teams and groups, having grown up with Facebook and Instant Messenger and interacting with friends through all kinds of electronic media. They respond equally well to wikis for working collaboratively on documents and presentations and sharing their source code on Sourceforge. One student said, "I now have a better understanding of what it is like to work with and contribute to a team of people, even when I may never meet them in person."

The HFOSS development process has no room for lone programmers working in isolation.

New Demographic

Computer science has not been broadly attractive at the undergraduate level, especially to women and other underrepresented groups. An April 2006 article in *Computing Research Association Bulletin*, based on data from the National Science Foundation and other sources, reported "[c]omputer science has the dubious distinction of being the only science field to see a fall in the share of its bachelor's degrees granted to women between 1983 and 2002" (http://www.cra.org/wp/index.php?p=83).

Attracting women and other underrepresented groups to computing remains a particularly challenging HFOSS Project objective. Only four women were enrolled in a 13-student introductory course in spring 2008, and for the summer 2008 internship program, only six out of 29 applicants were women. Of the 10 CPATH-funded summer interns only three were women, and two others were African-American. These numbers are not good, though they are somewhat better than the numbers in non-HFOSS computer science courses. For example, the fall 2008 CS1 courses offered at Connecticut College, Trinity College, and Wesleyan University included only 10 women and two African-American students out of a total of 69 students.

While this data is too sparse to support conclusions one way or the other regarding the appeal of HFOSS to women and other historically underrepresented groups, evaluations received from participating students suggest that the HFOSS approach has the potential to attract more women students to computing in the future. The responses from them suggest they speak positively about the project to their female friends. To help address this issue, we will, in summer 2010, extend the HFOSS Project to include a women's college and a traditionally black university. However, given the relatively small number of women and minorities who come to college with an interest in computing in the first place, the initiative may not solve the problem altogether; the solution, if there is one, may ultimately extend beyond the academy.

A widespread misconception about computing is that it is all about programming or coding. At most U.S. schools the introductory sequence focuses largely on teaching a programming language, further reinforcing this misconception. The HFOSS approach addresses it by contextualizing programming within a broader problem-solving activity. Being engaged in real-world projects with teams of developers, students see that programming is part of a complex, team-oriented, creative process that produces software to benefit society. Working closely with real clients, they see the need for transparent and secure code, extensive testing, and writing excellent user manuals and other supporting materials. They want to master these activities to improve their systems rather than step through mere academic exercises.

Another important HFOSS element is the ethic of sharing and collaboration. For this reason, the HFOSS Project teams students with one another, as well as with mentors, IT professionals, and HFOSS community members. The HFOSS development process has no room for lone programmers working in isolation.

Student feedback on these points reflects these observations. For example, one student said, "[this activity] shows how computer science can be a very helpful field of study than what we just know of it as programming in different programming languages." Another said, "[this activity] definitely changed my views of how effective software projects can be run. If we work collectively for the greater good, then we can get much more done."

The HFOSS Project has focused on individual courses and internships and only just begun to address how its approach might fit into an undergraduate curriculum. Reinforced throughout our experience is the longstanding view that computer science must be presented as a problem-solving discipline, and the more this value is built into the computing curriculum the more attractive it will be to a wider variety of bright students eager to solve problems. Georgia Tech and other institutions have begun exploring curricular models that contextualize programming within broader applications of computing (http://www.insidehighered.com/news/2006/09/26/gatech). The HFOSS approach would clearly complement such a model.

A common software industry complaint is that new computing graduates are strong on theory but lack practical understanding of the modern IT workplace. A common complaint

Logo of the Humanitarian FOSS Project (http://hfoss.org).

from academics is that IT professionals want colleges and universities to serve as training centers for their latest programming languages and software platforms. HFOSS addresses both by recruiting computing and IT professionals as advisers and mentors for its summer interns. For example, IT consultants from Accenture Corporation help mentor HFOSS students and serve as advisers in project management and other areas. Students appreciate the mentoring as they begin to understand the complexity of software development. They see that challenging problems rarely yield to "textbook" solutions and that the design process is often a protracted interaction between programmers and end users. One student said, "[this activity] definitely helped me understand more options of the IT profession. Now I know one more aspect of it, and how exciting it can be."

Portable, Sustainable Model

If the HFOSS model is to make a positive contribution to undergraduate computing curricula, it must continue to grow beyond the three campuses—Connecticut College, Trinity College, and Wesleyan University—where the Project began. During the past 18 months, with the support of the CPATH grant, we have seen evidence that such growth can be accomplished, as new HFOSS efforts began at Bowdoin College, Brunswick ME, and the University of Hartford, Hartford, CT. However, continued growth requires development of a supportive infrastructure and portable model that is easily adopted by other institutions.

Part of the effort to build a sustainable HFOSS model must include faculty development. Toward this end, we held outreach workshops for faculty at SIGCSE08 in Portland, OR, and CCSCNE08 in Staten Island, NY, (http://www.cs.trincoll.edu/hfoss/wiki/SIGCSE_2008_Workshop) to promote the HFOSS model as something worth trying. Feedback from workshop participants indicates that the humanitarian and FOSS aspects of the effort both have substantial appeal to computing faculty. However, despite this basic appeal, many challenges remain before more than a few other schools are able to integrate HFOSS into their computing curricula:

Faculty development. As with any new pedagogical endeavor, developing a new approach to teaching software design requires considerable initiative, time, and support. Faculty need time to learn new languages and tools and become active in the HFOSS community on their own before they are able to introduce HFOSS into their courses. To support this endeavor we are planning a summer training experience for faculty, similar to the week-long NSF-funded Chautauqua workshops (http://www.chautauqua.pitt.edu).

Software-tool support. Although FOSS software technology is free, creating a platform of FOSS tools to support a course or student project requires considerable time and effort. Faculty do not normally have time for downloading and installing software and making sure it works. One potential solution is a one-click installation that works on a variety of platforms. Another is for instructors to enlist such support among their universities' IT staff. The HFOSS project has begun to develop resources and processes to help, including a set of free and open Web-based resources, software tools, and other support materials (http://repository.hfoss.org).

Community development. Being involved in HFOSS means taking an active role in one or more HFOSS communities or projects, a process that can be somewhat bewildering and intimidating, especially for large well-established projects. We have identified and worked with communities and projects (described in the sidebar) that are accessible and welcoming. Sahana, OpenMRS, and InSTEDD are appreciative of student contributions and accepting of the compromises imposed by academic calendars and curricula. This summer we are working with the GNOME project on user-accessibility problems (http://projects.gnome.org/accessibility/). And a group of HFOSS students from several schools are currently working on the Portable Open Search and Identification Tool (POSIT), a disaster-management tool for the Google Android phone (http://code.google.com/p/posit-android/). All are ongoing projects that welcome contributions from faculty and students at other schools.

Cultural, institutional, curricular buy-in. Creating a new course or revising an existing one requires department support and approval. So the computer science academic community needs a more widespread and systematic discussion of how HFOSS might fit into the curriculum. Similarly, faculty development itself is not possible unless faculty and their departments recognize such engagement as an important form of community outreach and are therefore willing to invest the time and accept the complexity it requires. This may represent something of a cultural shift for some faculty.

Helping address these challenges, the HFOSS Project organized the first of what are planned to be an annual symposium on "Integrating FOSS into the Undergraduate Computing Curriculum" (http://www.hfoss.org/symposium09/). The March 2009 symposium's main goal was to bring together representatives from academia, industry, and the FOSS community to explore ways of integrating HFOSS into undergraduate teaching. The lively discussion that took place in Chattanooga, TN, helped identify a number of issues that stand in the way of more widespread adoption of the HFOSS model. For example, faculty participants identified a number of activities that could help them get involved, including summer training workshops and support for hosting open source code repositories.

Students see that challenging problems rarely yield to "textbook" solutions and that the design process is often a protracted interaction between programmers and end users.

Discussion focused on the kinds of support faculty and students would need to get started. One of the most promising ideas now being explored is establishment of a number of "HFOSS Chapters" whereby a faculty member and some students could take on a FOSS project (summer 2010). The software industry and FOSS-community representatives at the symposium expressed their eagerness to support the effort, including by helping train faculty to use FOSS tools and by providing "on ramps" to help faculty and students be integrated into the FOSS community.

To date, 15 additional schools have expressed interest in becoming HFOSS Chapters. Similarly, several more industry and FOSS-community supporters have volunteered to serve on the HFOSS Project steering committee and advisory board, including representatives from the GNOME project, Google, the Mozilla Foundation, RedHat, and Sun Microsystems.

Sustainability. No project can succeed in the long term without first encouraging the wide adoption of its methodologies and goals. But what would a sustainable model look like? In order to broadly influence undergraduate computing, high school and college students must be able to learn about FOSS and its humanitarian applications, thus requiring some kind of national organization and infrastructure to manage three functions:

- Funding internships (summer and, perhaps, academic year, too) to support student involvement in specific HFOSS projects;
- Funding a campaign to advertise HFOSS to prospective students, much as Teach for America and Habitat for Humanity advertise themselves; and
- Creating and managing an infrastructure whereby students are matched with specific HFOSS communities (such as Sahana and OpenMRS). The Google Summer of Code project, in which FOSS projects apply to Google for student-internship support, could serve as an adaptable model (http://code.google.com/soc/).

The hope is that the computing industry and FOSS communities embrace the potential value of HFOSS for computing students. In addition to revitalizing undergraduate computing education, a strong and diverse cohort of U.S. college graduates who come into the work force with FOSS experience will enrich the computing industry, along with the various FOSS communities.

Conclusion

Given the relative youth and scale of the HFOSS Project, it would be premature to make sweeping claims on its behalf. However, its ongoing objective is to systematically monitor its effects on undergraduate education to determine what would happen if students see computing as a discipline that develops software to help their friends and neighbors in need. Toward this end, the Project employs instruments and metrics, including student and faculty questionnaires, presentations at computing-education venues, and outside consultants from academic institutions and industry.

Though this evaluation is still preliminary, a number of promising signs have emerged.

First, HFOSS as a concept and methodology can indeed be introduced into the undergraduate computing curriculum. Our pedagogical experiments suggest that positive results are achievable through several approaches. For example, a general-education course can provide a coherent one-semester introduction to HFOSS techniques and to the broader cultural and societal effect of the HFOSS movement. Independent-study projects and internships provide a flexible venue through which students and faculty contribute to specific HFOSS projects in both the academic year and the summer. Upper-level software-engineering courses can be used to engage students in real-world HFOSS projects as part of their course work.

Second, feedback from faculty outreach activities, including the 2008 SIGCSE and CCSCNE workshops and the 2009 symposium, suggest there is significant faculty interest in integrating FOSS into the computing curriculum in many undergraduate institutions. Despite ongoing questions involving where, when, and how best to do it, the FOSS model is flexible enough to allow different institutions to answer these questions in ways that best suit their own programs.

Third, the students engaged thus far are attracted to the HFOSS concept for the opportunity to learn concepts, languages, and skills they don't see in other courses and for their interest in community service. Over the long run, these motivations promise to attract a wider range of capable students to computing, including more women and members of other underrepresented groups.

Fourth, student feedback suggests that engaging students in HFOSS projects helps foster a more constructive perception of the craft of programming and problem solving while generally reducing the computing-is-coding misconception. The ongoing HFOSS challenge is to spread this more positive perception across the entire undergraduate landscape. To some degree it will happen through word of mouth, as students share their positive HFOSS experiences with one another. But, as noted earlier, truly changing perceptions of computer science requires a concerted and sustained effort with broad support from the computing industry, the FOSS communities, primary and secondary schools, and society at large.

Finally, the HFOSS Project has expanded from its three initial schools, single corporate partner, and single software project into a vibrant community that today includes active faculty participants from eight U.S. colleges and universities (and expressed interest from many more), industry representatives from five IT corporations, and ongoing software-development projects with two local nonprofit organizations and five international FOSS communities. This growth—largely unplanned at the beginning of the Project—is indicative of a latent (inter)national interest in the HFOSS concept. If such expansion is sustained, it will help demonstrate that HFOSS can significantly affect the undergraduate computing curriculum, culture, and enrollment demographics.

Acknowledgments

HFOSS is supported by the National Science Foundation under grants #0722137, #0722134, and #0722199 and by the Mellon Foundation. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Mellon Foundation. C

References

1. Benkler, Y. and Nissenbaum, H. Commons-based peer production and virtue. *Journal of Political Philosophy 14*, 4 (Nov. 2006), 394–419.
2. Chopra, S. and Dexter, S. *Decoding Liberation: A Philosophical Investigation of Free Software.* Routledge, New York, 2007.
3. Ellis, H., Morelli, R., de Lanerolle, T., Damon, J., and Raye, J. Can humanitarian open source software development draw new students to CS? In *Proceedings of the 38th ACM SIGCSE Technical Symposium on Computer Science Education* (Covington, KY, Mar. 7–11), ACM Press, New York, 2007, 551–555.
4. Morelli, R. and de Lanerolle, T. FOSS 101: Engaging introductory students in the open source movement. In *Proceedings of the 40th ACM SIGSCE Technical Symposium on Computer Science Education* (Chattanooga, TN, Mar. 4–7), ACM Press, New York, 2009, 311–315.
5. Patterson, D. Rescuing our families, our neighbors, and ourselves. *Commun. ACM 48*, 11 (Nov. 2005), 29–31.
6. Patterson, D. President's letter: Computer science education in the 21st century. *Commun. ACM 49*, 3 (Mar. 2006), 27–30.
7. Raymond, E.S. The Cathedral and the Bazaar. O'Reilly Media, Sebastopol, CA, 2001; see also http://www.catb.org/~esr/writings/cathedralbazaar/.
8. Stallman, R. Viewpoint: Why 'open source' misses the point of free software. *Commun. ACM 52*, 6 (June 2009), 31–33.
9. Standish Group International. Free open source software is costing vendors $60 billion, new Standish Group International study finds. Standish Group International Press Release (Apr. 2008); http://www.marketwire.com/press-release/Standish-Group-International-844462.html.
10. Tucker, A. Teaching client-driven software development. *The Journal of Computing Sciences in College 24*, 4 (Apr. 2009), 29–39.

Ralph Morelli (ralph.morelli@trincoll.edu) is a professor of computer science at Trinity College, Hartford, CT.

Norman Danner (ndanner@wesleyan.edu) is an associate professor in the Department of Mathematics and Computer Science at Wesleyan University, Middletown, CT.

Allen Tucker (allen@bowdoin.edu) is the Anne T. and Robert M. Bass Professor Emeritus in the Computer Science Department at Bowdoin College, Brunswick, ME.

Heidi Ellis (hellis@wnec.edu) was a visiting professor at Trinity College at the time of this work and is now an associate professor and chair of Computer Science and Information Technology at Western New England College, Springfield, MA.

Gary Parker (parker@conncoll.edu) is an associate professor and chair of the Computer Science Department at Connecticut College, New London, CT.

Danny Krizanc (krizanc@wesleyan.edu) is a professor of computer science at Wesleyan University in Middletown, CT.

Trishan R. de Lanerolle (trishan.delanerolle@trincoll.edu) is project director of the Humanitarian FOSS project based at Trinity College, Hartford, CT.

Ozgur Izmirli (oizm@conncoll.edu) is an associate professor of computer science at Connecticut College, New London, CT.

DOI:10.1145/1536616.1536637

Satisfiability solvers can now be effectively deployed in practical applications.

BY SHARAD MALIK AND LINTAO ZHANG

Boolean Satisfiability

From Theoretical Hardness to Practical Success

THERE ARE MANY practical situations where we need to satisfy several potentially conflicting constraints. Simple examples of this abound in daily life, for example, determining a schedule for a series of games that resolves the availability of players and venues, or finding a seating assignment at dinner consistent with various rules the host would like to impose. This also applies to applications in computing, for example, ensuring that a hardware/software system functions correctly with its overall behavior constrained by the behavior of its components and their composition, or finding a plan for a robot to reach a goal that is consistent with the moves it can make at any step. While the applications may seem varied, at the core they all have variables whose values we need to determine (for example, the person sitting at a given seat at dinner) and constraints that these variables must satisfy (for example, the host's seating rules).

In its simplest form, the variables are *Boolean* valued (true/false, often represented using 1/0) and *propositional logic formulas* can be used to express the constraints on the variables.[15] In propositional logic the *operators* AND, OR, and NOT (represented by the symbols $\wedge$, $\vee$, and $\neg$ respectively) are used to construct formulas with variables. If x is a Boolean variable and f, f_1 and f_2 are propositional logic formulas (subsequently referred to simply as formulas), then the following recursive definition describes how complex formulas are constructed and evaluated using the constants 0 and 1, the variables, and these operators.

- x is a formula that evaluates to 1 when x is 1, and evaluates to 0 when x is 0
- $\neg f$ is a formula that evaluates to 1 when f evaluates to 0, and 0 when f evaluates to 1
- $f_1 \wedge f_2$ is a formula that evaluates to 1 when f_1 and f_2 both evaluate to 1, and

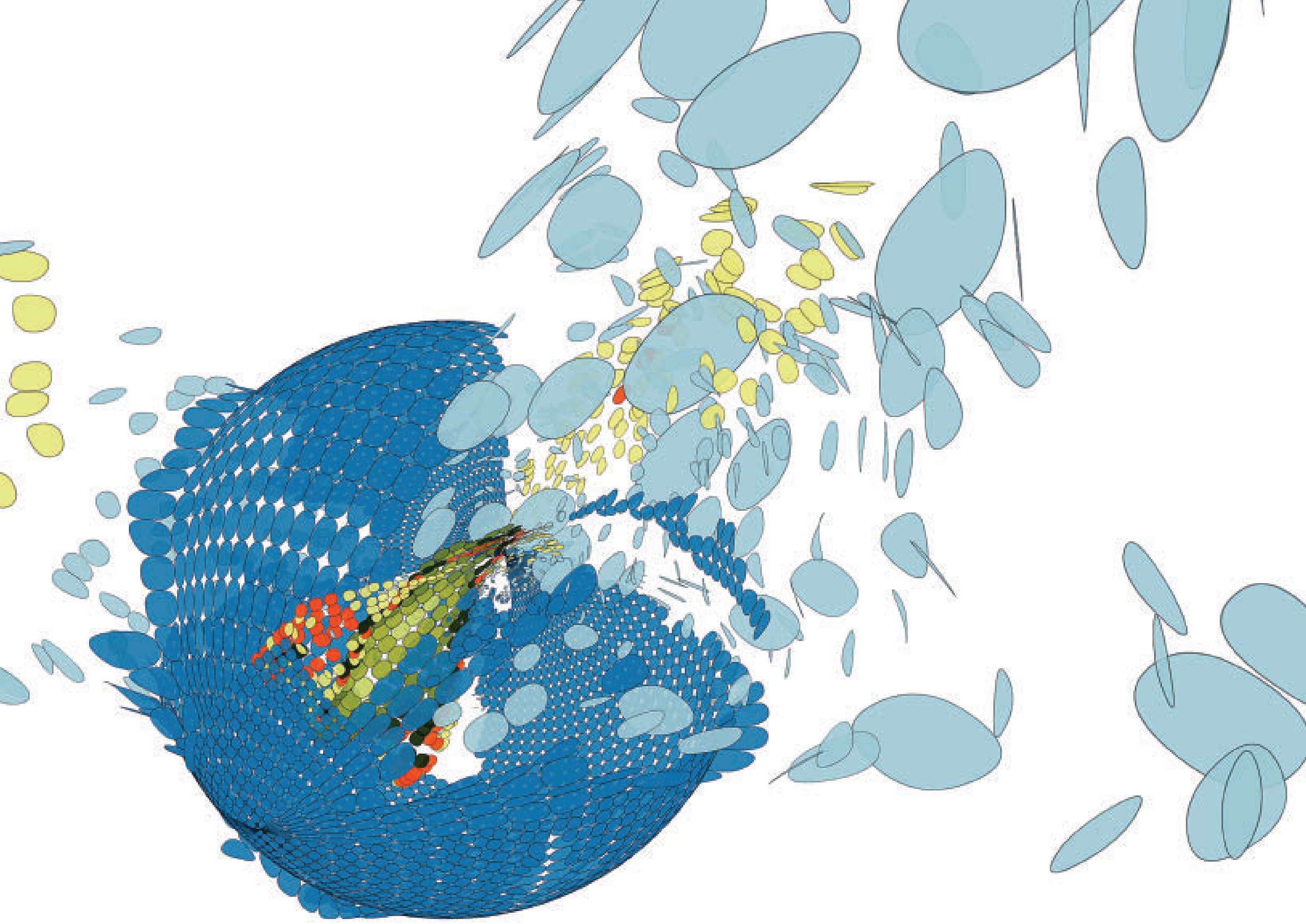


ILLUSTRATION BY GWEN VANHEE

evaluates to 0 if either f_1 or f_2 evaluate to 0

▸ $f_1 \vee f_2$ is a formula that evaluates to 0 when f_1 and f_2 both evaluate to 0, and evaluates to 1 if either f_1 or f_2 evaluate to 1

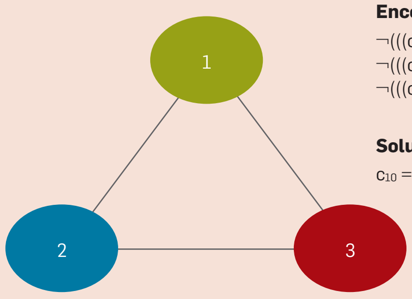
$(x_1 \vee \neg x_2) \wedge x_3$ is an example formula constructed using these rules. Given a valuation of the variables, these rules can be used to determine the valuation of the formula. For example: when $(x_1 = 0, x_2 = 0, x_3 = 1)$, this formula evaluates to 1 and when $(x_3 = 0)$, this formula evaluates to 0, regardless of the values of x_1 and x_2. This example also illustrates how the operators in the formula provide constraints on the variables. In this example, for this formula to be true (evaluate to 1), x_3 must be 1.

Boolean Satisfiability

A satisfying assignment for a formula is an assignment of the variables such that the formula evaluates to 1. It simultaneously satisfies the constraints imposed by all the operators in the formula. Such an assignment may not always exist. For example the formula $(\neg x_1 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$ cannot be satisfied by any of the four possible assignments 0/0, 0/1, 1/0, 1/1 to x_1 and x_2. In this case the problem is overconstrained. This leads us to a definition of the Boolean Satisfiability problem (also referred to as Propositional Satisfiability or just Satisfiability, and abbreviated as SAT): *Given a formula, find a satisfying assignment or prove that none exists.* This is the constructive version of the problem, and one used in practice. A simpler decision version, often used on the theoretical side, just needs to determine if there exists a satisfying assignment for the formula (a yes/no answer). It is easy to see that a solver for the decision version of the problem can easily be used to construct a solution to the constructive version, by solving a series of n decision problems where n is the number of variables in a formula.

Many constraint satisfaction problems dealing with non-Boolean variables can be relatively easily translated to SAT. For example, consider an instance of the classic graph coloring problem where an n-vertex graph needs to be checked for 4-colorability, that is, determining whether each vertex can be colored using one of four possible colors such that no two adjacent vertices have the same color. In this case, the variables are the colors $\{c_0, c_1, c_2, ..., c_{n-1}\}$ for the n vertices, and the constraints are that adjacent vertices must have different colors. For this problem the variables are not Boolean and the constraints are not directly expressed with the operators $\{\wedge, \vee, \neg\}$. However, the variables and constraints can be encoded into a propositional formula as follows. Two Boolean variables, c_{i0}, c_{i1}, are used in a two-bit encoding of the four possible values of the color for vertex i. Let i and j be adjacent vertices.

Figure 1. Encoding of graph coloring.



Encoding

$\neg(((c_{10} \wedge c_{20}) \vee (\neg c_{10} \wedge \neg c_{20})) \wedge ((c_{11} \wedge c_{21}) \vee (\neg c_{11} \wedge \neg c_{21}))) \wedge$
$\neg(((c_{10} \wedge c_{30}) \vee (\neg c_{10} \wedge \neg c_{30})) \wedge ((c_{11} \wedge c_{31}) \vee (\neg c_{11} \wedge \neg c_{31}))) \wedge$
$\neg(((c_{30} \wedge c_{20}) \vee (\neg c_{30} \wedge \neg c_{20})) \wedge ((c_{31} \wedge c_{21}) \vee (\neg c_{31} \wedge \neg c_{21})))$

Solution

$c_{10} = 0 \wedge c_{11} = 0 \wedge c_{20} = 0 \wedge c_{21} = 1 \wedge c_{30} = 1 \wedge c_{31} = 0$

The constraint $c_i \neq c_j$ is then expressed as $\neg((c_{i0} == c_{j0}) \wedge (c_{i1} == c_{j1}))$, here == represents equality and thus this condition checks that both bits in the encoding do not have the same value for i and j. Further, $(c_{i0} == c_{j0})$ can be expressed as $(c_{i0} \wedge c_{j0}) \vee (\neg c_{i0} \wedge \neg c_{j0})$, that is, they are both 1 or both 0. Similarly for $(c_{i1} == c_{j1})$. If we take the conjunction of the constraints on each edge, then the resulting formula is satisfiable if and only if the original graph coloring problem has a solution. Figure 1 illustrates an instance of the encoding of the graph coloring problem into a Boolean formula and its satisfying solution.

Encodings have been useful in translating problems from a wide range of domains to SAT, for example, scheduling basketball games,[40] planning in artificial intelligence,[20] validating software models,[17] routing field programmable gate arrays,[28] and synthesizing consistent network configurations.[29] This makes SAT solvers powerful engines for solving constraint satisfaction problems. However, SAT solvers are not always the best engines—there are many cases where specialized techniques work better for various constraint problems, including graph coloring (for example, Johnson et al.[19]). Nonetheless, it is often much easier and more efficient to use off-the-shelf SAT solvers than developing specialized tools from scratch.

One of the more prominent practical applications of SAT has been in the design and verification of digital circuits. Here, the translation to a formula is very straightforward. The functionality of digital circuits can be expressed as compositions of basic logic gates. A logic gate has Boolean input signals and produces Boolean output signals. The output of a gate can be used as an input to another gate. The functions of the basic logic gates are in direct correspondence to the operators $\{\wedge, \vee, \neg\}$. Thus various properties regarding the functionality of logic circuits can be easily translated to formulas. For example, checking that the values of two signals s_1 and s_2 in the logic circuit are always the same is equivalent to checking that their corresponding formulas f_1 and f_2 never differ, that is, $(f_1 \wedge \neg f_2) \vee (\neg f_1 \wedge f_2)$ is not satisfiable.

This technique can be extended to handle more complex properties involving values on sequences of signals, for example, a request is eventually acknowledged. For such problems, techniques that deal with temporal properties of the system, such as model checking, are used.[6] Modern SAT solvers have also been successfully applied for such tasks.[3,26] One of the main difficulties of applying SAT in checking such properties is to find a way to express the concept of "*eventually*." In theory, there is no tractable way to express this using propositional logic. However, in practice it is often good enough to just set a *bound* on the number of steps. For example, instead of asking whether a response to a request will eventually occur, we ask whether there will be a response within k clock cycles, where k is a small fixed number. Similar techniques have also been used in AI planning,[20] for example, instead of determining if a goal is reachable, we ask whether we can reach the goal in k steps. This unrolling technique has been widely adopted in practice, since we often only care about the behavior of the system within a small bounded number of steps.

Theoretical Hardness: SAT and NP-Completeness

The decision version of SAT, that is, determining if a given formula has a satisfying solution, belongs to the class of problems known as *NP-complete*.[8,12] An instance of any one of these problems can be relatively easily transformed into an instance of another. For example, both graph coloring and SAT are NP-complete, and earlier we described how to transform a graph coloring instance to a SAT instance.

All currently known solutions for NP-Complete problems, in the worst case, require runtime that grows exponentially with the size of the instance. Whether there exist subexponential solutions to NP-Complete problems is arguably the most famous open question in computer science.[a] Although there is no definitive conclusion, the answer is widely believed to be in the negative. This exponential growth in time complexity indicates the difficulty of scaling solutions to larger instances.

However, an important part of this characterization is "worst case." This holds out some hope for the "typical case," and more importantly the typical case that might arise in specific problem domains. *In fact, it is exactly the non-adversarial nature of practical instances that is exploited by SAT solvers.*

Solving SAT

Most SAT solvers work with a restricted representation of formulas in conjunctive normal form (CNF), defined as follows. A literal l is either a positive or a negative occurrence of a variable (for example, x or $\neg x$). A clause, c, is the OR of a set of literals, such as $(l_1 \vee l_2 \vee l_3 \ldots \vee l_n)$. A CNF formula is the AND of a set of clauses, such as $(c_1 \wedge c_2 \wedge c_3 \wedge\ c_m)$. An example CNF formula is:

$$(\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_4)$$

The restriction to CNF is an active choice made by SAT solvers as it enables their underlying algorithms. Further, this is not a limitation in terms of the formulas that can be handled. Indeed, with the addition of new auxiliary variables; it is easy to translate any formula into CNF with only a linear increase in size.[36] However, this representation is not used exclusively and there has been recent success with

a http://www.claymath.org/millennium/.

solvers for non-clausal representations (for example, NFLSAT[18]).

Most practically successful SAT solvers are based on an approach called systematic search. Figure 2 depicts the search space of a formula. The search space is a tree with each vertex representing a variable and the out edges representing the two decision choices for this variable. For a formula with n variables, there are 2^n leaves in the tree. Each path from the root to a leaf corresponds to a possible assignment to the n variables. The formula may evaluate to 1 or 0 at a leaf (colored green and red respectively). Systematic search, as the name implies, systematically searches the tree and tries to find a green leaf or prove that none exists.

The NP-completeness of the problem indicates that we will likely need to visit an exponential number of vertices in the worst case. The only hope for a practical solver is that by being smart in the search, almost all of the tree can be pruned away and only a minuscule fraction is actually visited in most cases. For an instance with a million variables, which is considered within the reach of modern solvers, the tree has 2^{10^6} leaves, and in reasonable computation time (about a day), we may be able to visit a billion (about 2^{30}) vertices as part of the search—a numerically insignificant fraction of the tree size!

Most search-based SAT solvers are based on the so called *DPLL approach* proposed by Davis, Logemann, and Loveland in a seminal *Communications* paper published in 1962.[9] (This research builds on the work by Davis and Putnam[10] and thus Putnam is often given shared credit for it.). Given a CNF formula, the DPLL algorithm first heuristically chooses an unassigned variable and assigns it a value: either 1 or 0. This is called branching or the *decision* step. The solver then tries to deduce the consequences of the variable assignment using deduction rules. The most widely used deduction rule is the *unit-clause* rule, which states that if a clause in the formula has all but one of its literals assigned 0 and the remaining one is unassigned, then the only way for the clause to evaluate to true, and thus the formula to evaluate to true, is for this last unassigned literal to be assigned to 1. Such clauses are called unit clauses and the forced assignments are called implications. This rule is applied iteratively until no unit clause exists. Note that this deduction is enabled by the CNF representation and is the main reason for SAT solvers preferring this form.

If at some point there is a clause in the formula with all of its literals evaluating to 0, then the formula cannot be true under the current assignment. This is called a *conflict* and this clause is referred to as a conflicting clause. A conflict indicates that some of the earlier decision choices cannot lead to a satisfying solution and the solver has to backtrack and try a different branch value. It accomplishes this by finding the most recent decision variable for which both branches have not been taken, flip its value, undo all variable assignments after that decision, and run the deduction process again. Otherwise, if no such conflicting clause exists, the solver continues by branching on another unassigned variable. The search stops either when all variables are assigned a value, in which case we have hit a green leaf and the formula is satisfiable, or when a conflicting clause exists when all branches have been explored, in which case the formula is unsatisfiable.

Consider the application of the algorithm to the formula shown in Figure 2. At the beginning the solver branches on variable x_1 with value 1. After branching, the first clause becomes unit and the remaining free literal $\neg x_2$ is implied to 1, which means x_2 must be 0. Now the second clause becomes unit and $\neg x_3$ is implied to 1. Then $\neg x_4$ is implied to 1 due to the third clause. At this point the formula is satisfied, and the satisfying assignment corresponds to the 8th leaf node from the left in the search tree. (This path is marked in bold in the figure.) As we can see, by applying the unit-clause rule, a single branching leads directly to the satisfying solution.

Many significant improvements in the basic DPLL algorithm have been

Figure 2. Search space of a formula.

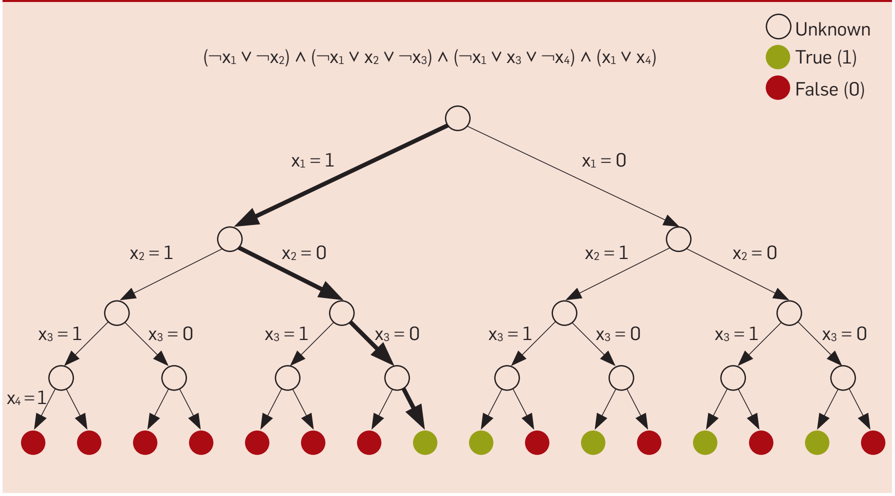
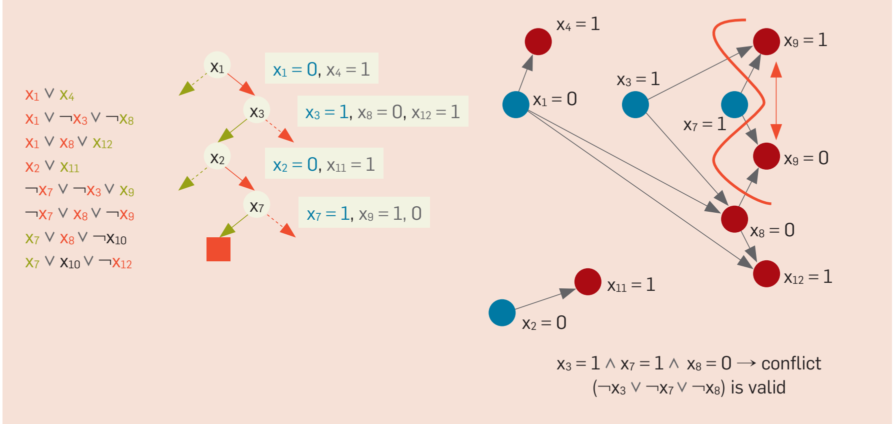


Figure 3. Conflict-driven learning and non-chronological backtracking.



proposed over the years. In particular, a technique called conflict-driven learning and non-chronological backtracking[2, 24] has greatly enhanced the power of DPLL SAT solvers on problem instances arising from real applications, and has become a key element of modern SAT solvers. The technique is illustrated in Figure 3. The column on the left lists the clauses in the example formula. The colors of the literals show the current assignments during the search (red representing 0, green 1, and black representing unassigned). The middle graph shows the branching and implications at the current point in the search. At each vertex the branching assignment is shown in blue and the implications in gray. The first branching is on x_1, and it implies x_4=1 (because of the first clause), the second branching is on x_3, and it implies x_8=0 and x_{12}=1 and so on. The right graph shows the implication relationships between variables. For example, x_4=1 is implied because of x_1=0, so there is a directed edge from node x_1=0 to node x_4=1. x_8=0 is implied because of both x_1=0 and x_3=1 (the red literals in the second clause), therefore, these nodes have edges leading to x_8=0.

After branching on x_7 and implying x_9=1 because of the 5th clause, we find that the 6th clause becomes a conflicting clause and the solver has to backtrack. Instead of flipping the last decision variable x_7 and trying x_7=0, we can learn some information from the conflict. From the implication graph, we see that there is a conflict because x_9 is implied to be both 1 and 0. If we consider a cut (shown as the orange line) separating the conflicting implications from the branching decisions, we know that once the assignments corresponding to the cut edges are made, we will end up with a conflict, since no further decisions are made. Thus, the edges that cross the cut are, in some sense, responsible for the conflict. In the example, x_3, x_7, and x_8 have edges cross the cut, thus the combination of x_3=1, x_7=1, and x_8=0 results in the conflict. We can learn from this and ensure that this assignment combination is not tried in the future. This is accomplished by recording the condition ($\neg x_3 \vee \neg x_7 \vee x_8$). This clause, referred to as a learned clause, can be added to the formula. While it is redundant in the sense that it is implied by the formula, it is nonetheless useful as it prevents search from ever making the assignment (x_3=1, x_7=1, x_8=0) again.

Further, because of this learned clause, $x_7 = 1$ is now implied after the second decision, and we can backtrack to this earlier decision level as the choice of $x_2 = 0$ is irrelevant to the current conflict. Since such backtracking skips branches, it is called non-chronological backtracking and helps prune away unsatisfiable parts of the search space.

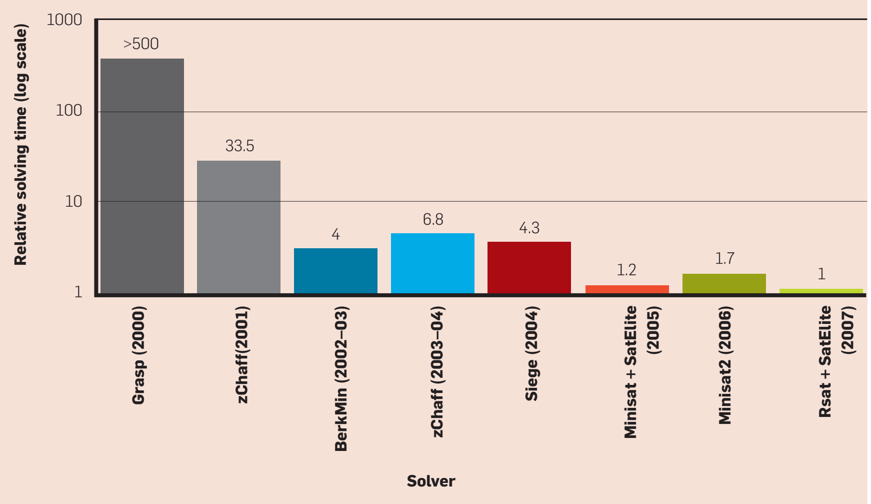
Recent Results

Recent work has exposed several significant areas of improvement now integral to modern SAT solvers.[22] The first deals with efficient implementation of the unit-clause rule using a technique called two-literal watching. The second area relates to improvements in the branching step by focusing on exhausting local sub-spaces before moving to new spaces. This is accomplished by placing increased emphasis on variables present in recently added conflict clauses. Another commonly used technique is random restart,[13] which periodically restarts the search while retaining the learned clauses from the current search to avoid being stuck in a search sub-space for too long. Other recent directions include formula preprocessing for clause and variable elimination,[11] considering algorithm portfolios that use empirical hardness models to choose among their constituent solvers on a per-instance basis[39] and using learning techniques to adjust parameters of heuristics.[16] With the advent of multicore processing, there is emerging interest in efficient multi-core implementations of parallel SAT solvers.[14]

The original Davis Putnam algorithm[10] based on resolution is often regarded as the first algorithm for SAT and has great theoretical and historical significance. However, this algorithm suffers from a space growth problem that makes it impractical. Reduced Ordered Binary Decision Diagrams (ROBDDs)[5] are a canonical representation of logic functions, that is, each function has a unique representation for a fixed variable ordering. Thus, ROBDDS can be used directly for SAT. However, ROBDDs also face space limitations with increasing instance size. Stålmarck's algorithm[35] uses breadth-first search instead of depth-first search as in DPLL, and has been shown to be practically useful. Its performance relative to DPLL based solvers is unclear as public versions of efficient implementations of Stålmarck's algorithm are not available due to its proprietary nature.

When represented in CNF, SAT can be regarded as a discrete optimization problem with the objective to maximize the number of satisfied clauses. If this max value is equal to the total number of clauses, then the instance is satisfiable. Many discrete optimization techniques have been explored in the SAT context, including simulated annealing,[33] tabu search,[25] neural networks,[34] and genetic algorithms.[23]

Figure 4. Speedup of SAT solvers in recent years.



A variation of the optimization approach, first proposed in the early 1990s, solves SAT using local search (for example, GSAT[31]). The algorithm first randomly selects a value for each variable, and calculates how many clauses are satisfied. If not all clauses are satisfied, it repeatedly flips the value of a variable to increase the number of the clauses satisfied. If no such variable is available, it accepts a decrease in the objective function by either flipping a random variable, or restarting from a fresh set of variable assignments. This is accelerated further, by confining the flips to literals in clauses not satisfied by the current assignment.[30] This simple algorithm, when carefully implemented, is surprisingly effective on certain classes of SAT instances. Unfortunately, this algorithm is incomplete in the sense that while it may be able to find an assignment for a satisfiable SAT instance, it cannot prove an instance to be unsatisfiable. More recently, incomplete solvers based on a technique called survey propagation[4] have been found to be very effective for certain classes of SAT instances and have attracted much attention in the theory community.

The Role of Benchmarks

It is important to note the role of practical benchmarks in the development of modern SAT solvers. These benchmarks are critical in tuning the solvers to various classes of practical instances (that is, instances generated from real-world applications). While we do not have deep insight into how these solvers exploit the special structure found in these instances, we do know that the structure is critical in our ability to tackle them. (There exists some recent work that provides initial insights into the effect of structure on DPLL search.[37,38]) Experimental research in SAT solvers has been enabled in large part by benchmarks put forward collectively by the research community, and the challenge in the form of a SAT solver competition that is held regularly with the International Conference on Theory and Applications of Satisfiability Testing (SAT).[b] The community has also benefited from the SATLive portal, which has provided widespread dissemination of links to SAT articles and software.[c]

b http://www.satcompetition.org/.

One of the more prominent practical applications of SAT has been in the design and verification of digital circuits. The functionality of digital circuits can be expressed as compositions of basic logic gates.

Figure 4 provides some data on the improvements in SAT solvers at the SAT Competition in recent years.[d] It plots the relative solving times for a set of solvers developed over the last 10 years. This includes solvers that placed first in the industrial benchmarks category of the SAT competitions. The solvers were run on a set of benchmarks from hardware and software verification (not used in the competitions).[32] This is normalized to the best solver in the 2007 competition (RSAT with the SatElite preprocessor). The slow-down of the Grasp solver is a lower bound, since it could not complete some of the benchmarks in the 10,000-second time limit. While this study is limited to a specific set of benchmarks, it is indicative of the progress in SAT solvers since 2000.

Industrial Impact

SAT solvers are maturing to the point that developers are using them in a range of application domains, much like mathematical programming tools or linear equation solvers. Early use of SAT was seen in planning in artificial intelligence with practical use in space exploration.[27] Recent increases in the capacity of commercial solvers has enabled widespread use in the electronic design automation (EDA) industry as the reasoning engine behind verification and testing tools such as automatic test pattern generators,[21] equivalence checkers, and property checkers. SAT-based bounded model checkers have been used in industrial microprocessor verification.[7] More recently, SAT has also been used in tools for software verification and debugging, for example, industrial verification of device drivers using SAT-based model checking,[e] as well as SAT-based static analysis.[f] Outside of verification and testing, SAT techniques have also been applied in configuration management such as resolving software package dependencies.[g]

c http://www.satlive.org/.
d Provided by Sanjit Seshia, UC Berkeley.
e http://www.microsoft.com/whdc/DevTools/tools/SDV.mspx.
f http://www.coverity.com/index.html.
g http://news.opensuse.org/2008/06/06/sneak-peeks-at-opensuse-110-package-management-with-duncan-mac-vicar/.

Beyond SAT

The success with SAT solvers has emboldened researchers to consider problems related to, but more difficult than SAT. The most promising of these is Satisfiability Modulo Theories (SMT) that has received significant attention in recent years.

In SAT, the variables are assumed to be constrained only by the clauses in the formula. SMT extends SAT by considering the case when the variables may be connected by one or more underlying theories. For example, consider the formula $(x_1 \wedge \neg x_2 \wedge x_3)$. This formula is clearly satisfiable with $(x_1 = 1, x_2=0, x_3=1)$. However, if x_1, x_2 and x_3 represent the following relationships among the real variables y_1 and y_2:

x_1: $y_1 < 0$
x_2: $y_1 + y_2 < 1$
x_3: $y_2 < 0$

Then, in fact, there is no assignment to y_1 and y_2 for which $(x_1 = 1, x_2=0, x_3=1)$, i.e., y_1 and y_2 cannot be both negative and their sum at least one. Thus, the original formula is unsatisfiable given this underlying relationship. In this example, the specific theory used to determine the validity of a satisfying assignment is Linear Real Arithmetic. Emerging SMT solvers can incorporate reasoning for a range of theories such as Linear Integer Arithmetic, Difference Logic, Arrays, Lists, Uninterpreted Functions and many others, including their combinations.[1] The theoretical difficulty depends on the specific theories considered. SMT is seeing rapid progress and initial commercial use in software verification.

Conclusion

The success with SAT has led to its widespread commercial use in certain domains such as design and verification of hardware and software systems. There is even a sense in parts of the computer science community that this problem has been successfully tamed in practice. This is probably too optimistic a view. There are still enough instances that are difficult for current solvers, and it is unclear if they will be able to handle the change in scale/nature of instances from yet unseen domains. However, there is definitely a sense of confidence that we will be able to continue to strengthen our solvers.

Given its theoretical hardness, the practical success of SAT has come as a surprise to many in the computer science community. The combination of strong practical drivers and open competition in this experimental research effort created enough momentum to overcome the pessimism based on theory. Can we take these lessons to other problems and domains? C

References

1. Barrett, C., Sebastiani, R., Seshia, S., and Tinelli, C. Satisfiability modulo theories. A. Biere, H. van Maaren, T. Walsh, Eds. *Handbook of Satisfiability 4*, 8 (2009), IOS Press, Amsterdam.
2. Bayardo R., and Schrag, R. Using CSP look-back techniques to solve real-world SAT instances. *National Conference on Artificial Intelligence*, 1997.
3. Biere, A., Cimatti, A., Clarke, E. M., and Zhu, Y. Symbolic model checking without BDDs. *Tools and Algorithms for the Analysis and Construction of Systems*, 1999.
4. Braunstein, A., Mezard, M., and Zecchina, R. Survey propagation: An algorithm for Satisfiability. *Random Structures and Algorithms 27* (2005), 201–226.
5. Bryant, R.E. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers C-35* (1986), 677–691.
6. Clarke, E.M., Grumberg, O., and Peled, D.A. *Model Checking.* MIT Press, Cambridge, MA, 1999.
7. Copty, F., Fix, L., Fraer, R., Giunchiglia, E., Kamhi, G., Tacchella, A., and Vardi, M.Y. Benefits of bounded model checking at an industrial setting. *Proceedings of the 13th International Conference on Computer-Aided Verification*, 2001.
8. Cook, S.A. The complexity of theorem-proving procedures. *Third Annual ACM Symposium on Theory of Computing*, 1971.
9. Davis, M., Logemann, G., and Loveland, D. A machine program for theorem proving. *Comm. ACM 5* (1962), 394–397.
10. Davis, M., and Putnam, H. A computing procedure for quantification theory. *JACM 7* (1960), 201–215.
11. Eén, N., and Biere, A. Effective preprocessing in SAT through variable and clause elimination. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, 2005.
12. Garey, M.R., and Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979
13. Gomes, C. P., Selman, B., and Kautz, H. Boosting combinatorial search through randomization. In *Proceedings of National Conference on Artificial Intelligence* (Madison, WI, 1998).
14. Hamadi, Y., Jabbour, S., and Sais, L. ManySat: Solver description. Microsoft Research, TR-2008-83.
15. Huth, M. and Ryan, M. *Logic in Computer Science: Modeling and Reasoning about Systems.* Cambridge University Press, 2004.
16. Hutter, F., Babic, D., Hoos, H.H., and Hu, A. J. Boosting verification by automatic tuning of decision procedures. In *Proceedings of the International Conference on Formal Methods in Computer-Aided Design*, (Austin, TX, Nov. 2007).
17. Jackson, D., and Vaziri, M., Finding bugs with a constraint solver. In *Proceedings of the International Symposium on Software Testing and Analysis* (Portland, OR, 2000).
18. Jain, H. Verification using satisfiability checking, predicate abstraction, and Craig interpolation. Ph.D. Thesis, Carnegie-Mellon University, School of Computer Science, CMU-CS-08-146, 2008.
19. Johnson, D.S., Mehrotra, A., and Trick, M. A. Preface: Special issue on computational methods for graph coloring and its generalizations. *Discrete Applied Mathematics 156*, 2; Computational Methods for Graph Coloring and its Generalizations. (Jan. 15, 2008), 145–146.
20. Kautz, H. and Selman, B. Planning as satisfiability. *European Conference on Artificial Intelligence*, 1992.
21. Larrabee, T. Test pattern generation using Boolean satisfiability. *IEEE Transactions on Computer-Aided Design* (Jan. 1992) 4–15.
22. Madigan, M.W., Madigan, C.F., Zhao, Y., Zhang, L., and Malik, S. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Conference on Design Automation.* (New York, NY, 2001).
23. Marchiori, E. and Rossi, C. A flipping genetic algorithm for hard 3-SAT problems. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Orlando, FL, 1999), 393–400.
24. Marques-Silva, J.P, and Sakallah, K.A. Conflict analysis in search algorithms for propositional satisfiability. *IEEE International Conference on Tools with Artificial Intelligence*, 1996.
25. Mazure, B., Sas L., and Grgoire, E., Tabu search for SAT. In *Proceedings of the 14th National Conference on Artificial Intelligence* (Providence, RI, 1997).
26. McMillan, K.L., Applying SAT methods in unbounded symbolic model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification*. Lecture Notes In Computer Science 2404 (2002). Springer-Verlag, London, 250–264.
27. Muscettola, N., Pandurang Nayak, P., Pell, B., and Williams, B.C. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence 103*, 1–2, (1998), 5–47.
28. Nam, G.-J., Sakallah, K. A., and Rutenbar, R.A. Satisfiability-based layout revisited: Detailed routing of complex FPGAs via search-based Boolean SAT. *International Symposium on Field-Programmable Gate Arrays* (Monterey, CA, 1999).
29. Narain, S., Levin, G., Kaul, V., Malik, S. Declarative infrastructure configuration and debugging. *Journal of Network Systems and Management, Special Issue on Security Configuration.* Springer, 2008.
30. Selman, B., Kautz, H.A., and Cohen, B. Noise strategies for improving local search. In *Proceedings of the 12th National Conference on Artificial Intelligence* (Seattle, WA, 1994). American Association for Artificial Intelligence, Menlo Park, CA, 337–343.
31. Selman, B., Levesque, H., and Mitchell, D. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence*, (1992) 440–446.
32. Seshia, S.A., Lahiri, S.K., and Bryant, R.E. A hybrid SAT-based decision procedure for separation logic with uninterpreted functions. In *Proceedings of the 40th Conference on Design Automation* (Anaheim, CA, June 2–6, 2003). ACM, NY, 425–430; http://doi.acm.org/10.1145/775832.775945.
33. Spears, W.M. Simulated annealing for hard satisfiability problems. *Cliques, Coloring and Satisfiability, Second DIMACS Implementation Challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* D.S. Johnson and M.A. Trick, Eds. American Mathematical Society (1993), 533–558.
34. Spears, W. M. A NN algorithm for Boolean satisfiability problems. In *Proceedings of the 1996 International Conference on Neural Networks*, 1121–1126.
35. Stålmarck, G. A system for determining prepositional logic theorems by applying values and rules to triplets that are generated from a formula. U.S. Patent Number 5276897, 1994.
36. Tseitin, G. On the complexity of derivation in propositional calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part 2* (1968), 115–125. Reprinted in *Automation of reasoning vol. 2.* J. Siekmann and G. Wrightson, Eds. Springer Verlag, Berlin, 1983, 466–483.
37. Williams, R., Gomes, C., and Selman, B. Backdoors to typical case complexity. In *Proceedings. of the 18th International Joint Conference on Artificial Intelligence* (2003), 1173–1178.
38. Williams, R., Gomes, C., and Selman, B. On the connections between heavy-tails, backdoors, and restarts in combinatorial search. In *Proceedings of the International Conference on Theory and Applications of Satisfiability Testing*, 2003.
39. Xu, L., Hutter, F., Hoos, H. H., Leyton-Brown, K. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research 32*, (2008), 565–606.
40. Zhang, H. Generating college conference basketball schedules by a SAT solver. In *Proceedings of the 5th International Symposium on Theory and Applications of Satisfiability Testing.* (Cincinnati, OH, 2002).

Sharad Malik (sharad@princeton.edu) is a professor in the Department of Electrical Engineering at Princeton University, Princeton, NJ.

Lintao Zhang (lintaoz@microsoft.com) is a researcher at Microsoft Research Asia, Beijing, China.

DOI:10.1145/1536616.1536638

Technical Perspective
Maintaining Quality in the Face of Distributed Development

By James Herbsleb

IT WAS A problem that should not have taken three weeks to solve. But the tester was in Germany and the developer was in England. The documentation claimed that if a function was called from a command line with particular parameters, it would return values of particular state variables. If the operator simply entered blank, it would return the values of all the variables. It was this last option that was causing the grief. Entering blank just returned garbage, insisted the tester. The developer couldn't duplicate the problem, and after three weeks of frustrating emails and phone conversations, the developer hopped on an airplane to Germany. A few seconds after sitting down beside the tester, he observed the tester enter the characters "b-l-a-n-k" and hit return, rather than just hitting return by itself. Mystery solved.

This is just one anecdote, but it is emblematic of how most everyone these days thinks of globally distributed development. Whatever benefits might be realized, one is also likely to encounter a steady diet of frustration, delay, misunderstandings, mistakes, and cross-purposes. And there is no shortage of research supporting these intuitions. Study after study has provided rich descriptions of the variety of problems encountered, and quantified their cumulative effects. Delay due to multi-site projects has received particular attention, as the numerous small holdups, quite salient to developers, accumulate into a significant burden.

But enough of this doom and gloom, say Bird et al. In a study of Vista, a very large, widely distributed project, the authors take a close look at the impact of geography on software quality, and, to everyone's surprise—including the authors—they find little or none. Binaries developed within a single building, or across boundaries including different buildings, campuses, or even continents, have virtually the same rates of failures, after controlling for other factors. Here, finally, is some encouraging news for those going global.

The study is important not just for the overall result, but also for the care that was taken in achieving that result. The authors take full advantage of the rare opportunity provided by their impressive data set. They have data from a company directory that allows them to consider many levels of geographic distribution, rather than the coarser binary distributed-versus-collocated distinction typical of previous research. Moreover, their sample size is sufficient to give credibility to their negative results. As a rule, because of the way that statistical tests are used in an experimental context, a negative result (when the predicted differences are not observed) is difficult to interpret. Maybe the effect does not exist, or maybe it does exist but the study was not sensitive enough to observe it. But with a sufficiently large sample and a carefully conducted study, however, one can have confidence that if a substantial effect existed it is highly likely that it would have been detected. This is such a study, and the negative results are convincing.

In a study of Vista, a very large, widely distributed project, the authors take a close look at the impact of geography on software quality, and, to everyone's surprise—including the authors—they find little or none.

The authors also go to great pains to rule out other possible explanations that could cloud the results. For example, maybe only relatively simple binaries are developed in distributed fashion. If that were the case, then perhaps distributed development efforts achieved a dead heat in quality with collocated development only because the distributed teams had an easier task. The authors did a careful analysis of the differences between their distributed and collocated binaries, and found virtually no differences. It appears the comparison was meaningful—apples to apples, so to speak. One small exception to this was a weak tendency for distributed work to involve more people, an intriguing parallel to an earlier finding my collaborators and I encountered when analyzing multisite delay, as the authors pointed out. This hint that somehow more people get pulled into the work when the work spans sites seems worthy of further investigation.

Finally, the authors carefully revisit the literature, and point out that several of the conditions shown in the past to disrupt distributed projects were not present in the project they studied. The sites used a consistent tool set, for example, shared common schedules, and had ample opportunity to overcome cultural differences. This rich description of the context of the project will be very helpful for future researchers who may find different results. It will help us eventually to sort through the potential causes of quality problems as case studies accumulate.

The following paper is an important contribution, a terrific read, and an elegant example of bringing scientific methods to bear on a problem of both theoretical and practical concern.

James Herbsleb (jdh@cs.cmu.edu) is a professor of computer science at Carnegie Mellon University, Pittsburgh, PA.

DOI:10.1145/1536616.1536639

Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista

By Christian Bird, Nachiappan Nagappan, Premkumar Devanbu, Harald Gall, and Brendan Murphy

Abstract

Existing literature on distributed development in software engineering, and other fields discusses various challenges, including cultural barriers, expertise transfer difficulties, and communication and coordination overhead. Conventional wisdom, in fact, holds that distributed software development is riskier and more challenging than collocated development. We revisit this belief, empirically studying the overall development of Windows Vista and comparing the post-release failures of components that were developed in a distributed fashion with those that were developed by collocated teams. We found a negligible difference in failures. This difference becomes even less significant when controlling for the number of developers working on a binary. Furthermore, we also found that component characteristics (such as code churn, complexity, dependency information, and test code coverage) differ very little between distributed and collocated components. Finally, we examine the software process used during the Vista development cycle and examine how it may have mitigated some of the difficulties of distributed development introduced in prior work in this area.

1. INTRODUCTION

Globally distributed software development is an increasingly common strategic response to issues such as skill set availability, acquisitions, government restrictions, increased code size, cost and complexity, and other resource constraints.[4, 9] In this paper, we examine development that is globally distributed, but completely within Microsoft. This style of *global development* within a single company is to be contrasted with *outsourcing*, which involves multiple companies. It is widely believed that distributed collaboration involves challenges not inherent in collocated teams, including delayed feedback, restricted communication, less shared project awareness, difficulty of synchronous communication, inconsistent development and build environments, and lack of trust and confidence between sites.[20] While there are studies that have examined the delay associated with distributed development and the direct causes for them,[11] there has been much less attention (see e.g., Rammasubbu and Balan[21]) to the effect of distributed development on software quality in terms of post-release failures.

In this paper, we use historical development data from the implementation of Windows Vista, along with post-release failure information, to empirically evaluate the hypothesis that globally distributed software development leads to more failures. We focus on post-release failures at the level of individual executables and libraries (which we refer to as binaries) shipped as part of the operating system and use the IEEE definition of a failure as "the inability of a system or component to perform its required functions within specified performance requirements." Post-release failures are the most costly to companies in terms of reputation and marketshare.

Using geographical and commit data for the developers that worked on Vista, we divide the Vista binaries into those developed by (a) distributed and (b) collocated teams; we then examine the distribution of post-release failures in both populations. Binaries are classified as developed in a distributed manner if at least 25% of the commits came from locations other than where binary's owner resides. We find that there is a small (around 10%) increase in the number of failures of binaries written by distributed teams (hereafter referred to as distributed binaries) over those written by collocated teams (collocated binaries). However, when controlling for team size, the difference becomes negligible. In order to see if only smaller, less complex, or less critical binaries are chosen for distributed development (which could explain why distributed binaries have approximately the same number of failures), we examined many relevant properties of these binaries, but found no difference between distributed and collocated binaries. We present our methods and findings in this paper.

2. MOTIVATION AND CONTRIBUTIONS

Distributed software development is a general concept that can be operationalized in various ways. Development may be distributed along many dimensions with various distinctive characteristics.[8] There are key questions that should be clarified when discussing a distributed software project. Who or what is distributed and at what level? Are people or the artifacts distributed? Are people dispersed individually or dispersed in groups?

It is important to consider the way that developers and other entities are distributed. The distribution can be across

A previous version of this article appeared in *Proceedings of the 31st International Conference on Software Engineering* (May 2009).

geographical, organizational, temporal, or stakeholder boundaries.[14] A scenario involving one company outsourcing work to another will certainly differ from another, where multiple, distributed teams work within the same company. A recent special issue of IEEE Software focused on globally distributed development, but the majority of the papers dealt with *offshoring* relationships between separate companies and *outsourcing*, which are likely very different from distributed sites within the same company.[2, 5, 6] Even within a company, the development may or may not span organizational structure at different levels. Do geographical locations span the globe, including multiple time zones, languages, and cultures or are they simply in different cities of the same state or nation?

We are interested in studying the effect of globally distributed software development within the *same* company, because there are many issues involved in outsourcing that are independent of geographical distribution (e.g. expertise finding, different process, and an asymmetric relationship). Our main motivation is to confirm or refute the notion that global software development leads to more failures within the context of our setting.

To our knowledge, this is the first large scale distributed development study that considers distributed development *within an organization*. This study augments the current body of knowledge and differs from prior studies by making the following contributions:

1. We examine distributed development at multiple levels of separation (building, campus, continent, etc.).
2. We examine a large scale software development effort, composed of thousands of binaries and thousands developers.
3. We examine complexity and maintenance characteristics of the distributed and collocated binaries to check for inherent differences that might influence post-release quality.
4. Our study examines a project in which all sites involved are part of the same company and have been using the same process and tools for years.

There is a large body of theory describing the difficulties inherent in distributed development. We summarize them here.

Communication suffers due to a lack of unplanned and informal meetings.[10] Engineers do not get to know each other on a personal basis. Synchronous communication becomes less common due to time zone and language barriers. Even when communication is synchronous, the communication channels, such as conference calls or instant messaging, are less rich than face to face and collocated group meetings. Developers may take longer to solve problems because they lack the ability to step into a neighboring office to ask for help. They may not even know the correct person to contact at a remote site.

Coordination breakdowns occur due to this lack of communication and lower levels of group awareness.[1, 3] When managers must manage across large distances, it becomes more difficult to stay aware of peoples' task and how they are interrelated. Different sites often use different tools and processes which can also make coordinating between sites difficult.

Diversity in operating environments may cause management problems.[1] Often there are relationships between the organization doing development and external entities such as governments and third party vendors. In a geographically dispersed project, these entities will be different based on location (e.g., national policies on labor practices may differ between the United States and India).

Distance can reduce team cohesion[20] in groups collaborating remotely. Eating, sharing an office, or working late together to meet a deadline, all contribute to a feeling of being part of a team. These opportunities are diminished by distance.

Organizational and national cultural barriers may complicate globally distributed work.[4] Coworkers must be aware of cultural differences in communication behaviors. One example of a cultural difference within Microsoft became apparent when a large company meeting was originally (and unknowingly) planned on a major national holiday for one of the sites involved.

Based on these prior observations and an examination of the hurdles involved in globally distributed development we expect that difficulties in communication and coordination will lead to an increase in the number of failures in code produce by distributed teams over code from collocated teams. We formulate our testable hypothesis formally.

H1: Binaries that are developed by teams of engineers that are distributed will have more post-release failures than those developed by collocated engineers.

We are also interested to see if the binaries that are distributed differ from their collocated counterparts in any significant ways. It is possible that managers, aware of the difficulties mentioned above, may choose to develop simpler, less frequently changing, or less critical software in a distributed fashion. We therefore present our second hypothesis.

H2: Binaries that are distributed will be less complex, experience less code churn, and have fewer dependencies than collocated binaries.

3. RELATED WORK

There is a wealth of literature in the area of globally distributed software development. It has been the focus of multiple special issues of IEEE Software, workshops at ICSE and the International Conference on Global Software Engineering. Here we survey important work in the area, including both studies and theory of globally distributed work in software development.

There have been a number of experience reports for globally distributed software development projects at various companies including Siemens,[13] Alcatel,[7] Motorola,[1] Lucent,[10] and Philips.[15]

3.1. Effects on bug resolution

In an empirical study of globally distributed software development,[11] Herbsleb and Mockus examined the time to resolution of Modification Requests (MRs) in two departments of Lucent working on distinct network elements for a telecommunication system. The average time needed to complete a "single-site" MR was 5 days versus 12.7 for "distributed." When controlling for other factors such as number of people working on an MR, how diffused the changes are across the code base, size of the change, and severity, the effect of being distributed was no longer significant. They hypothesize that large and/or multi-module changes are both more time consuming and more likely to involve multiple sites. These changes require more people, which introduce delay. They conclude that distributed development indirectly introduces delay due to correlated factors such as team size and breadth of changes required.

Thanh et al.[19] examined the effect of distributed development on delay between communications and time to resolution of work items in IBM's Jazz project, which was developed at five globally distributed sites. While Kruskal–Wallis tests showed a statistically significant difference in resolution times for items that were more distributed, the Kendall Tau correlations of time to resolution and time between comments with number of sites were extremely low (below 0.1 in both cases). This indicates that distributed collaboration does not have a strong effect.

Herbsleb and Mockus[12] formulate an empirical theory of coordination in software engineering and test hypotheses based on this theory. They precisely define software engineering as requiring a sequence of decisions associated with a project. Each decision constrains the project and future decisions in some way, until all choices have been made, and the final product does or does not satisfy the requirements. It is therefore important that only feasible decisions (those which will lead to a project that does satisfy the requirements) be made. They present a coordination theory, and develop testable hypotheses regarding productivity, measured as number of MRs resolved per unit time. They find that (a) people who are assigned work from many sources have lower productivity, and that (b) MRs that require work in multiple modules have a longer cycle time than those which require changes to just one.

Unlike the above papers, our study focuses on the effect of distributed development on defect *occurrence*, rather than on defect *resolution time*.

3.2. Effects on quality and productivity

Diomidis Spinellis examined the effect of distributed development on productivity, code style, and defect density in the FreeBSD code base.[23] He measured the geographical distance between developers, the number of defects per source file, as well as productivity in terms of number of lines committed per month. A correlation analysis showed that there is little, if any, relationship between geographic distance of developers and productivity and defect density. It should be noted that this is a study of open source software which is, by its very nature, distributed and has a very different process model from commercial software.

Cusick and Prasad[5] examined the practices used by Wolters Kluwer Corporate Legal Services when outsourcing software development tasks and present their model for deciding if a project is offshorable and how to manage it effectively. Their strategies include keeping communication channels open, using consistent development environments and machine configurations, bringing offshore project leads onsite for meetings, developing and using necessary infrastructure and tools, and managing where the control and domain expertise lies. They also point out that there are some drawbacks that are difficult to overcome and should be expected such as the need for more documentation, more planning for meetings, higher levels of management overhead, and cultural differences. This work was based on an offshoring relationship with a separate vendor and not collaboration between two entities within the same company. We expect that the challenges faced in distributed development may differ based on the type of relationship between distributed sites.

Ramasubbu and Balan[21] examined the relationship between the *dispersion* (a measure of geographic dispersion) of a project and its *development productivity* and *conformance quality*. They gathered information from 42 projects over 2 years and found that projects that had more dispersion also had lower levels of productivity and conformance quality, though the effects were strongly mitigated through quality management approaches. In their study, productivity and quality were measured on a project basis between different projects, while our study examines characteristics of components within one large software project, which arguably provides better control over possibly confounding project-specific factors.

Our study examines distributed development in the context of one commercial entity, which differs greatly from both open source projects and outsourcing relationships.

3.3. Issues and solutions

In his paper on global software teams,[3] Carmel categorizes project risk factors into four categories that act as centrifugal forces that pull global projects apart. These are

- Loss of communication richness
- Coordination breakdowns
- Geographic dispersion
- Cultural differences

In 2001, Battin et al.[1] discussed the challenges and their solutions relative to each of Carmel's categories in a large scale project implementing the 3G Trial (Third Generation Cellular System) at Motorola. By addressing these challenges in this project, they found that globally distributed software development did not increase defect density, and in fact, had lower defect density than the industrial average. Table 1 lists the various locations, the size of the code developed at those locations, and their defect density. They summarize the key actions necessary for success with global development in order of importance:

- Use Liaisons
- Distribute entire things for entire life cycle
- Plan to accommodate time and distance

Carmel and Agarwal[4] present three tactics for alleviating distance in global software development, each with examples, possible solutions, and caveats:

- Reduce intensive collaboration.
- Reduce national and organizational cultural distance.
- Reduce temporal distance.

Table 1. Locations, code size, and defect density from Motorola's 3G trial project for each site.

Development Locations	Code Size (KLOC C/C++)	Defect Density (Defects/KLOC)
Beijing, China	57	0.7
Arlington Heights, USA	54	0.8
Arlington Heights, USA	74	1.3
Tokyo, Japan	56	0.2
Bangalore, India	165	0.5
Singapore	45	0.1
Adelaide, Australia	60	0.5

Nagappan et al. investigated the influence of organizational structure on software quality in Windows Vista.[18] They found a strong relationship between how development is distributed across the organizational structure and number of post-release failures in binaries shipped with the operating system. Along with other organizational measures, they measured the level of code ownership by the organization that the binary owner belonged to, the number of organizations that contributed at least 10% to the binary, and the organizational level of the person whose reporting engineers perform more than 75% of the edits. Our paper complements this study by examining geographically, rather than organizationally distributed development.

4. METHODS AND ANALYSIS

In this section, we describe our methods of gathering data for our study and the analysis methods used to evaluate our hypotheses regarding distributed development in Windows Vista.

4.1. Data collection

Windows Vista is a large commercial software project involving a few thousand developers. It comprises thousands of binaries (defined as individual files containing machine code such as executables or a libraries) with a source code base of tens of millions LOC. Developers were distributed across 59 buildings and 21 campuses in Asia, Europe, and North America. Vista was developed completely in-house without any outsourced elements.

Our data focuses on three properties: code quality, geographical location, and code ownership. Our measure of code quality is post-release failures, since these matter most to end-users, cost the most to fix, and affect product and company reputation. These failures are recorded for the 6 months following the release of Vista at the binary level.

The *geographical location* of each software developer at Microsoft is obtained from the people management software at the time of release to manufacturing of Vista. This data includes the building, campus, region, country, and continent information. While some developers occasionally move, it is standard practice at Microsoft to keep a software engineer at one location during an entire product cycle. Most of the developers of Vista didn't move during the observation period.

Finally we gathered the number of commits made by each engineer to each binary. We remove build engineers from the analysis because their changes are broad, but not substantive. Many files have fields that need to be updated prior to a build, but the actual source code is not modified. By combining this data with developer geographical data, we determine the level of distribution of each binary and categorize these levels into a hierarchy. Microsoft practices a strong code ownership development process. We found that on average, 49% of the commits for a particular binary can be attributed to one engineer. Although we are basing our analysis on data from the development phase, in most cases, this is indicative of the distribution that was present during the design phase as well.

We categorized the distribution of binaries into the following geographic levels. Our reasoning behind this classification is explained below.

Building: Developers who work in the same building (and often the same floor) will enjoy more face to face and informal contact. A binary classified at the building level may have been worked on by developers on different floors of the same building.

Cafeteria: Several buildings share a cafeteria. One cafeteria services between one and five nearby buildings. Developers in different, but nearby buildings, may "share meals" together or meet by chance during meal times. In addition, the typically shorter geographical distance facilitates impromptu meetings.

Campus: A campus represents a group of buildings in one location. For instance, in the United States, there are multiple campuses. Some campuses are located in the same city. It is easy to travel between buildings on the same campus by foot while travel between campuses (even in the same city) requires a vehicle.

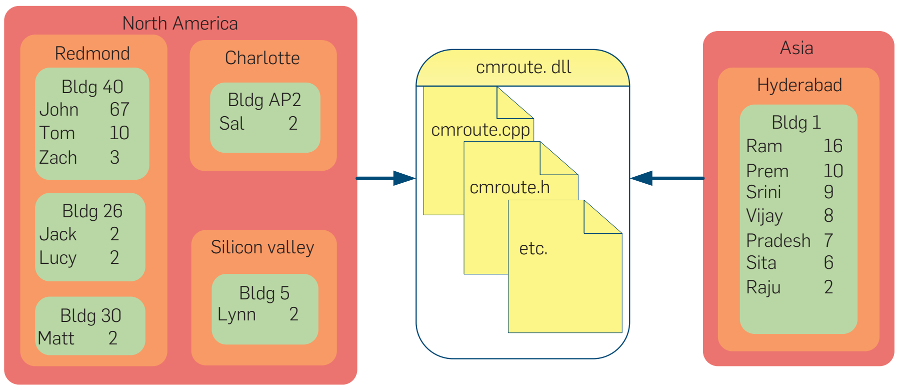
Locality: We use localities to represent groups of geographically proximate campuses. For instance, the Seattle locality contains all of the campuses in western Washington. One can travel within a locality by car on day trips, but travel between localities often requires air travel and multi-day trips. Also, all sites in a particular locality operate in the same time zone, making coordination and communication within a locality easier than between localities.

Continent: All of the locations on a given continent fall into this category. We choose to group at the continent level rather than the country level because Microsoft has offices in Vancouver Canada and we wanted those to be grouped together with other west coast sites (Seattle to Vancouver is less than 3 h by road). If developers are located in the same continent, but not the same locality, then it is likely that cultural similarities exists, but they operate in different time zones and rarely speak face to face.

World: Binaries developed by engineers on different continents are placed in this category. This level of geographical distribution means that face to face meetings are rare and synchronous communication such as phone calls or online chats are hindered by time differences. Also, cultural and language differences are more likely.

For every level of geographical dispersion there are more than two entities from the lower level within that level. That is, Vista was developed in more that three continents, localities, etc. Each binary is assigned the lowest level in the hierarchy from which at least 75% of the commits were made. Thus,

Figure 1: Hierarchy of distribution levels in Windows Vista.



if engineers residing in one region make at least 75% of the commits for a binary, but there is no campus that accounts for 75%, then the binary is categorized at the region level. This threshold was chosen based on results of prior work on development distributed across organizational boundaries that is standardized across Windows.[18] Figure 1 illustrates the geographic distribution of commits to an actual binary (with names anonymized). To assess the sensitivity of our results to this selection and address any threats to validity we performed the analysis using thresholds of 60%, 75%, 90%, and 100% with consistently similar results.

Note that whether a binary is distributed or not is orthogonal to the actual location where it was developed. Some binaries that are classified at the building level were developed entirely in a building in Hyderabad, India while others were owned in Redmond, Washington.

Figure 2 illustrates the hierarchy and shows the proportion of binaries that fall into each category. Note that a majority of binaries have over 75% of their commits coming from just one building. The reason that so few binaries fall into the continent level is that the Unites States is the only country which contains multiple localities. Although the proportion of binaries categorized above the campus level is barely 10%, this still represents a sample of over 380 binaries; enough for a strong level of statistical power.

We initially examined the number of binaries and distribution of failures for each level of our hierarchy. In addition, we divided the binaries into "distributed" and "collocated" categories in five different ways using, each time using a different level shown in Figure 2 (e.g., one split categorizes building and cafeteria level binaries as collocated and the rest as distributed). These categorizations are used to determine if there is a level of distribution above which there is a significant increase in the number of failures. The results from analysis of these dichotomized data sets were consistent in nearly all respects. We therefore present the results of the first data set and point out deviations between the data sets where they occurred.

4.2. Experimental analysis

In order to test our hypothesis about the difference in code quality between distributed and collocated development, we examined the distribution of the number of post-release failures per binary in both populations. Figure 3 shows histograms of the number of bugs for distributed and collocated binaries. Absolute numbers are omitted from the histograms for confidentiality, but the horizontal and vertical scales are the same for both histograms. A visual inspection indicates that although the mass is different, with more binaries categorized as collocated than distributed, the distribution of failures are very similar.

A Mann–Whitney test was used to quantitatively measure the difference in means because the number of failures was not normally distributed.[16] The difference in means is statistically significant, but small. While the average number of failures per binary is higher when the binary was distributed, the actual magnitude of the increase is only about 8%. In a prior study by Herbsleb and Mockus,[11] time to resolution of

Figure 2. Commits to the library cmroute.dll. For clarity, location of anonymized developers is shown only in terms of continents, regions, and buildings.

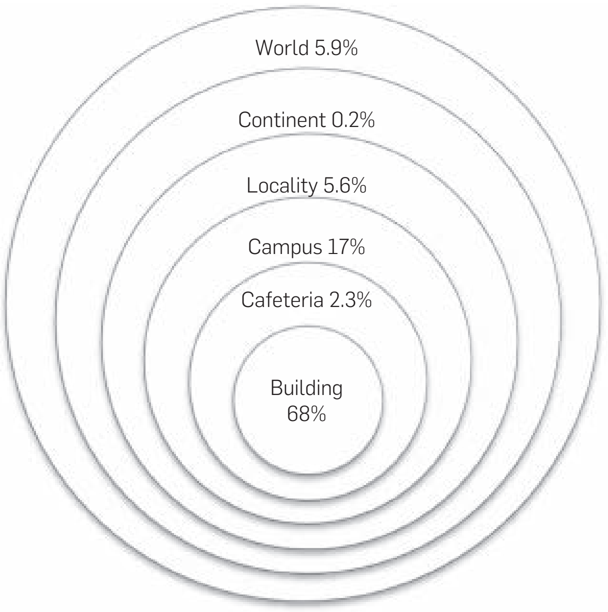
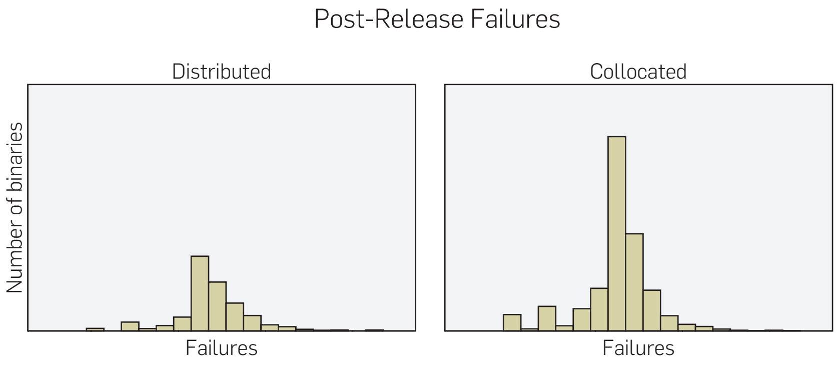


Figure 3. Histograms of the number of failures per binary for distributed (left) and collocated (right) binaries. Although numbers are not shown on the axes, the scales are the same in both histograms.



MRs was positively correlated with the level of distribution of the participants. After further analysis, they discovered that the level of distribution was not significant when controlling for the number of people participating. We performed a similar analysis on our data.

We used linear regression to examine the effect of distributed development on number of failures. Our initial model contained only the binary variable indicating whether or not the binary was distributed. The number of developers working on a binary was then added to the model and we examined the coefficients in the model. In these models, *distributed* is a binary variable indicating if the binary is distributed and *numdevs* is the number of developers that worked on the binary. We show here the results of analysis when splitting the binaries at the regions level. The *F*-statistic and *p* value show how likely the null hypothesis (the hypothesis that the predictor variable has no effect on the response variable) is. We give the percentage increase in failures when the binaries are distributed based on the parameter values. As *numdevs* is only included in the models to examine effect of distribution when controlling for number of developers we do not include estimates or percentage increase.

Model 1. *F* Statistic = 12.43, *p* < .0005

Variable	% Increase	Standard Error	Significance
(Constant)		0.30	*p* < .0005
distributed	9.2%	0.31	*p* < .0005

This indicates that on average, a distributed binary has 9.2% more failures than a collocated binary. However, the result changes then controlling for the number of developers working on a binary.

Model 2. *F* Statistic = 720.74, *p* < .0005

Variable	% Increase	Standard Error	Significance
(Constant)		0.25	*p* < .0005
distributed	4.6%	0.25	*p* = .056
numdevs		0.00	*p* < .0005

We performed this analysis on all five splits of the binaries (one at each level as shown in Figure 2). The estimates for *distributed* coefficient for all models were below 17%, and dropped even further to below 9% when controlling for number of developers (many were below this value, but the numbers cited are upper bounds). In addition, the effect of *distributed* in models that accounted for the number of developers was only statistically significant when dividing binaries at the continents level. In concrete terms, this indicates that a binary contributed to by 20 developers in Redmond will have relatively the same number of defects as one that has commits from 20 developers around the world.

We also used linear regression to examine the effect of the level of distribution on the number of failures of a binary. Since the level of distribution is a nominal variable that can take on six different values, we encode it into five binary variables. The variable *diff_buildings* is 1 if the binary was distributed among different buildings that all were served by the same cafeteria and 0 otherwise, etc. The percentage increase for each *diff* represents the increase in failures relative to binaries that are developed by engineers in the same building.

Model 3. *F* Statistic = 25.48, *p* < .0005

Variable	% Increase	Standard Error	Significance
(Constant)		0.09	*p* < .0005
diff_buildings	15.1%	0.50	*p* < .0005
diff_cafeterias	16.3%	0.21	*p* < .0005
diff_campuses	12.6%	0.35	*p* < .0005
diff_localities	2.6%	1.47	*p* = .824
diff_continents	−5.1%	0.31	*p* = .045

The parameter estimates of the model indicate that binaries developed by engineers on the same campus served by different cafeterias have, on average, 16% more post-release failures than binaries developed in the same building. Interestingly, the change in number of failures is quite low for those developed in multiple regions and continents. However, when controlling for development team size, only binaries categorized at the levels of different cafeterias and different campuses show a statistically significant increase in failures over binaries developed in the same building. Even so, the actual effects are relatively minor (4% and 6%, respectively).

Model 4. *F* Statistic = 242.73, $p < .0005$

Variable	% Increase	Standard Error	Significance
(Constant)		0.09	$p < .0005$
diff_buildings	2.6%	0.42	$p = .493$
diff_cafeterias	3.9%	0.18	$p = .016$
diff_campuses	6.3%	0.29	$p = .019$
diff_localities	8.3%	1.23	$p = .457$
diff_continents	−3.9%	0.26	$p = .101$
numdevs		0.00	$p < .0005$

Two important observations can be made from these models. The first is that the variance explained by the predictor variables as measured in the adjusted R^2 value (not shown) for the built models rises from 2% and 4% (models 1 and 3) to 33% (models 2 and 4) when adding the number of developers. The second is that when controlling for the number of developers, not all levels of distribution show a significant effect, but the increase in post-release failures for those that do is minimal with values at or below 6%. To put this into perspective, a binary with 4 failures if collocated would have 4.24 failures if distributed. Although our response variable is different from Herbsleb and Mockus, our findings are consistent with their result that when controlling for the number of people working on a development task, distribution does not have a large effect. Based on these results, we are unable to reject the null hypothesis and H1 is not confirmed.

This leads to the surprising conclusion that in the context in which Windows Vista was developed, teams that were distributed wrote code that had virtually the same number of post-release failures as those that were collocated.

4.3. Differences in binaries

One possible explanation for this lack of difference in failures could be that distributed binaries are smaller, less complex, have fewer dependencies, etc. Although the number of failures changes only minimally when the binaries are distributed, we are interested in the differences in characteristics between distributed and collocated binaries. This was done to determine if informed decisions were made about which binaries should be developed in a distributed manner. For instance, prior work has shown that the number of failures is highly correlated with code complexity and number of dependencies.[17, 24] Therefore, it is possible that only less complex binaries or those with less dependents were chosen for distribution in an effort to mitigate the perceived dangers of distributed development.

We gathered metrics for each of the binaries in an attempt to determine if there is a difference in the nature of binaries that are distributed. These measures fall into five broad categories.

Size and Complexity: Our code size and complexity measures include number of independent paths through the code, number of functions, classes, parameters, blocks, lines, local and global variables, and cyclomatic complexity. From the call graph we extract the fan in and fan out of each function. For object oriented code we include measures of class coupling, inheritance depth, the number of base classes, subclasses and class methods, and the number of public, protected, and private data members and methods. All of these are measured as totals for the whole binary and as maximums on a function or class basis as applicable.

Code Churn: As measures of code churn we examine the change in size of the binary, the frequency of edits and the churn size in terms of lines removed, added, and modified from the beginning of Vista development until release to manufacturing.

Test Coverage: The number of blocks and arcs as well as the block coverage and arc coverage are recorded during the testing cycle for each binary.

Dependencies: Many binaries have dependencies on one another (in the form of method calls, data types, registry values that are read or written, etc.). We calculate the number of direct incoming and outgoing dependencies as well as the transitive closure of these dependencies. The depth in the dependency graph is also recorded.

People: We include a number of statistics on the people and organizations that worked on the binaries. These include all of the metrics in our prior organizational metrics paper[18] such as the number of engineers that worked on the binary.

We began with a manual inspection of the 20 binaries with the least and 20 binaries with the most number of post-release failures in both the distributed and collocated categories and examined the values of the metrics described above. The only discernible differences were metrics relative to the number of people working on the code, such as team size.

Metric	Average Value	Correlation	Significance
Functions	895.86	0.114	$p < .0005$
Complexity	4603.20	0.069	$p < .0005$
Churn Size	53430.00	0.057	$p = .033$
Edits	63.82	0.134	$p < .0005$
Indegree	13.04	−0.024	$p = .363$
Outdegree	9.67	0.100	$p < .0005$
Number of Devs	21.55	0.183	$p < .0005$

We evaluated the effect of these metrics on level of distribution in the entire population by examining the spearman rank correlation of distribution level of binaries (not limited to the "top 20" lists) with the code metrics. Most metrics had correlation levels below 0.1 and the few that were above that level, such as number of engineers, never exceeded 0.25. Logistic regression was used to examine the relationship of the development metrics with distribution level. The increase in classification accuracy between a naive model including no independent variables and a stepwise refined model with 15 variables was only 4%. When removing data related to people that worked on the source, the refined model's accuracy only improved 2.7% from the naive model. We include the average values for a representative sample of the metrics along with a spearman rank correlation with the level of distribution for the binaries and the significance of the correlation. Although the *p*-values are quite low, the magnitude of the correlation is small. This is attributable to the very large sample of binaries (over 3,000).

We conclude that there is no discernible difference in the measured metrics between distributed and collocated binaries.

5. DISCUSSION

We have presented an unexpected, but encouraging result: it is possible to conduct in-house globalized distributed development without adversely impacting quality. It is certainly important to understand why this occurred and how this experience can be repeated in other projects and contexts. To prime this future endeavor, we make some observations concerning pertinent practices that have improved communication, coordination, team cohesion, etc., and reduced the impact of differences in culture and business context. These observations come from discussions with management as well as senior and experienced developers.

Relationship between Sites: Much of the work on distributed development examines outsourcing relationships.[2, 6] Others have looked at strategic partnerships between companies or scenarios in which a foreign remote site was acquired.[10] These create situations where relationships are asymmetric. Engineers at different sites may feel competitive or may for other reasons be less likely to help each other. In our situation, all sites have existed and worked together on software for many years. There is no threat that if one site performs better, the other will be shut down. The pay scale and benefits are equivalent at all sites in the company.

Cultural Barriers: In a study of distributed development within Lucent at sites in Great Britain and Germany, Herbsleb and Grinter[10] found that significant national cultural barriers existed. These led to a lack of trust between sites and misinterpreted actions due to lack of cultural context. This problem was alleviated when a number of engineers (liaisons) from one site visited another for an extended period of time. Battin et al.[1] found that when people from different sites spent time working together in close proximity, many issues such as trust, perceived ability and delayed response to communication requests were assuaged.

A similar strategy was used during the development of Vista. Development occurred mostly in the United States (predominantly in Redmond) and Hyderabad, India. In the initial development phases, a number of engineers and executives left Redmond to work at the Indian site. Many of these people had 10+ years within Microsoft, and understood the company's development process. In addition, the majority of these employees were originally from India, removing one key challenge from globally distributed work. These people acted as facilitators, information brokers, recommenders, and cultural liaisons[4] and had already garnered a high level of trust and confidence from the engineers in the United States. Despite constituting only a small percent of the Indian workforce, they helped to reduce both organizational and national cultural distances.[4]

Communication: Communication is the single most referenced problem in globally distributed development. Face to face meetings are difficult and rare and people are less likely to communicate with others that they don't know personally. Distributed sites are also more likely to use asynchronous communication channels such as email which introduce a task resolution delay.[22]

The Vista developers made heavy use of synchronous communication daily. Employees took on the responsibility of staying at work late or arriving early for a status conference call on a rotating basis, changing the site that needed to keep odd hours every week. Keeping in close and frequent contact increases the level of awareness and the feeling of "teamness."[1, 4] This also helps to convey status and resolve issues quickly before they escalate. Engineers also regularly traveled between remote sites during development for important meetings.

Consistent Use of Tools: Both Battin[1] and Herbsleb and Mockus[11] cite the importance of the configuration management tools used. In the case of Motorola's project, a single, distributed configuration management tool was used with great success. At Lucent, each site used their own management tools, which led to an initial phase of rapid development at the cost of cumbersome integration work toward the end. Microsoft employs the use of one configuration management and builds system throughout all of its sites. Every engineer is familiar with the same source code management tools, development environment, documentation method, defect tracking system, and integration process. The integration process for code is incremental, allowing problems to surface early.

End to End Ownership: Distributed ownership is a problem with distributed development. When an entity fails, needs testing, or requires a modification, it may not be clear who is responsible for performing the task or assigning the work. Battin mentions ownership of a component for the entire life cycle as one of three critical strategies when distributing development tasks. While binaries were committed to from different sites during the implementation phase, Microsoft practices strong code ownership. One developer is clearly "in control" of a particular piece of code from design, through implementation, and into testing and maintenance. Effort is made to minimize the number of ownership changes.

Common Schedules: All of the development that we examined was part of one large software project. The project was not made up of distributed modules that shipped separately. Rather, Vista had a fixed release date for all parties and milestones were shared across all sites. Thus all engineers had a strong interest in working together to accomplish their tasks within common time frames.

Organizational Integration: Distributed sites in Microsoft do not operate in organizational isolation. There is no top level executive at India or China that all the engineers in those locations report to. Rather, the organizational structure spans geographical locations at low levels. It is not uncommon for engineers at multiple sites to have a common direct manager. This, in turn, causes geographically dispersed developers to be more integrated into the company and the project. The manager can act as a facilitator between engineers who may not be familiar with one another and can also spot problems due to poor coordination earlier than in an organizational structure based purely on geography, with less coupling between sites. Prior work has shown that organizationally distributed development dramatically affects the number of post-release defects.[18]. This organizational integration across geographic boundaries reconciles the results of that work with the conclusions reached in this study. Organizational culture is fairly consistent across geography because the same process has been used in all locations of the company for some time.

6. THREATS TO VALIDITY

Construct Validity: The data collection on a system the size of Windows Vista is automated. Metrics and other data were collected using production level quality tools and we have no reason to believe that there were large errors in measurement.
Internal Validity: In Section 5 we listed observations about the distributed development process used at Microsoft. While we have reason to believe that these alleviate the problems associated with distributed development, a causal relationship has not been empirically shown. Further study is required to determine to what extent each of these practices actually helps. In addition, although we attempted an exhaustive search of differences in characteristics between distributed a collocated binaries, it is possible that they differ in some way not measured by our analysis in Section 4.3.
External Validity: It is unclear how well our results generalize to other situations. We examine one large project and there is a dearth of literature that examines the effect of distributed development on post-release failures. We have identified similarities in Microsoft's development process with other successful distributed projects, which may indicate important principles and strategies to use. There are many ways in which distributed software projects may vary and the particular characteristics must be taken into account. For instance, we have no reason to expect that a study of an outsourced project would yield the same results as ours.

7. CONCLUSION

In our study we divide binaries based on the level of geographic dispersion of their commits. We studied the post-release failures for the Windows Vista code base and concluded that distributed development has little to no effect. We posit that this negative result is a significant finding as it refutes, at least in the context of Vista development, conventional wisdom and widely held beliefs about distributed development. When coupled with prior work,[1, 11] our results support the conclusion that there are scenarios in which distributed development can work for large software projects. Based on earlier work,[18] our study shows that organizational differences are much stronger indicators of quality than geography. An organizationally compact but geographically distributed project would be better than a geographically local, organizationally distributed project.

We have presented a number of observations about the development practices at Microsoft which may mitigate some of the hurdles associated with distributed development, but no causal link has been established. There is a strong similarity between these practices and those that have worked for other teams in the past[1] as well as solutions proposed in other work.[10] Directly examining the effects of these practices is an important direction for continued research in globally distributed software development. Devanbu and Bird acknowledge that their work is in part supported by the National Science Foundation, under Grant NSF-SOD 0613949. C

References

1. Battin, R.D., Crocker, R., Kreidler, J., Subramanian, K. Leveraging resources in global software development. *IEEE Softw. 18*, 2 (Mar./Apr. 2001), 70–77.
2. Bhat, J.M., Gupta, M., Murthy, S.N. Overcoming requirements engineering challenges: Lessons from offshore outsourcing. *IEEE Softw. 23*, 6 (Sept./Oct. 2006), 38–44.
3. Carmel, E. *Global Software Teams: Collaborating across Borders and Time Zones*. Prentice Hall, 1999.
4. Carmel, E., Agarwal, R. Tactical approaches for alleviating distance in global software development. *IEEE Softw. 2*, 18 (Mar./Apr. 2001), 22–29.
5. Cusick, J., Prasad, A. A practical management and engineering approach to offshore collaboration. *IEEE Softw. 23*, 5 (Sept./Oct. 2006), 20–29.
6. Desouza, K.C., Awaza, Y., Baloh, P. Managing knowledge in global software development efforts: Issues and practices. *IEEE Softw. 23*, 5 (Sept./Oct. 2006), 30–37.
7. Ebert, C., Neve, P.D. Surviving global software development. *IEEE Softw. 18*, 2 (2001), 62–69.
8. Gumm, D.C. Distribution dimensions in software development projects: a taxonomy. *IEEE Softw. 23* (2006), 545–551.
9. Herbsleb, J. Global software engineering: the future of socio-technical coordination. *International Conference on Software Engineering*, 2007, 188–198.
10. Herbsleb, J., Grinter, R. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Softw*. (1999).
11. Herbsleb, J., Mockus, A. An empirical study of speed and communication in globally distributed software development. *IEEE Trans. Softw. Eng.* (2003).
12. Herbsleb, J.D., Mockus, A. Formulation and preliminary test of an empirical theory of coordination in software engineering. In *Proceedings of 11th International Symposium on Foundations of Software Engineering* (2003).
13. Herbsleb, J.D., Paulish, D.J., Bass, M. Global software development at siemens: Experience from nine projects. In *Proceedings of the 27th International Conference on Software Engineering* (2005), ACM, 524–533.
14. Holmstrom, H., Conchuir, E., Agerfalk, P., Fitzgerald, B. Global software development challenges: A case study on temporal, geographical and socio-cultural distance. *Proceedings of the IEEE International Conference on Global Software Engineering* (2006), 3–11.
15. Kommeren, R., Parviainen, P. Philips experiences in global distributed software development. *Empirical Softw. Eng. 12*, 6 (2007), 647–660.
16. Mann, H.B., Whitney, D.R. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat. 18*, 1 (1947), 50–60.
17. Nagappan, N., Ball, T., Zeller, A. Mining metrics to predict component failures. In *Proceedings of the International Conference on Software Engineering* (2006).
18. Nagappan, N., Murphy, B., Basili, V. The influence of organizational structure on software quality: An empirical case study. In *Proceedings of the 30th International Conference on Software Engineering* (2008).
19. Nguyen, T., Wolf, T., Damian, D. Global software development and delay: Does distance still matter? In *Proceedings of the International Conference on Global Software Engineering* (2008).
20. Olson, G.M., Olson, J.S. Distance matters. *Hum. Comp. Interact. 15*, 2/3 (2000), 139–178.
21. Rammasubbu, N., Balan, R. Globally distributed software development project performance: An empirical analysis. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering* (2007), ACM, New York, NY, USA, 125–134.
22. Sosa, M., Eppinger, S., Pich, M., McKendrick, D., Stout, S., Manage, T., Insead, F. Factors that influence technical communication in distributed product development: An empirical study in the telecommunications industry. *IEEE Trans. Eng. Manage. 49*, 1 (2002), 45–58.
23. Spinellis, D. Global software development in the freebsd project. In *GSD '06: Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner* (Shanghai, China, 2006), 73–79.
24. Zimmermann, T., Nagappan, N. Predicting defects using network analysis on dependency graphs. In *Proceedings of the International Conference on Software Engineering* (2008).

Christian Bird (cabird@ucdavis.edu), University of California, Davis, Davis, CA.

Nachiappan Nagappan (nachin@microsoft.com), Microsoft Research, Redmond, WA.

Premkumar Devanbu (ptdevanbu@ucdavis.edu), University of California, Davis, Davis, CA.

Harald Gall (gall@ifi.uzh.ch), University of Zurich, Zurich, Switzerland.

Brendan Murphy (bmurphy@microsoft.com), Microsoft Research, Cambridge, England.

DOI:10.1145/1536616.1536640

Technical Perspective
Where the Chips May Fall

By Sachin S. Sapatnekar

CIRCUIT SPEED, OR timing, is one of the most crucial specifications on the performance of an integrated circuit. Since on-chip delays depend on the values of process parameters that drive transistor and wire characteristics, timing has traditionally been evaluated at several "corners," corresponding to process parameter settings that could result in nominal, best-case, or worst-case delays. The traditional approach to circuit design has been to build chips that work correctly at these extreme-case process corners, thereby guard-banding them against process variations.

This corner-based approach has been the mainstay of timing analysis for decades, but is on a collision course with the Moore's Law trend that doubles the number of transistors on a chip every 18–24 months. This doubling has primarily been achieved by making transistor widths and heights smaller by a factor of about $1/\sqrt{2}$, and appropriately scaling the wires, in each technology generation (as a result, the area of a chip is halved and twice the number of devices can fit in the same area). However, the characteristics of these smaller devices are more difficult to control: for instance, the fabrication process introduces variations in the physical dimensions of devices and wires; at tiny geometries, the dopant atoms within a transistor can no longer be considered uniformly distributed; and so on.

While some of these variations are predictable and can be incorporated relatively easily into conventional timing analysis, others must be modeled as uncorrelated or correlated random variations. If all of these random variations on a die were perfectly correlated, they could be evaluated at a worst-case or best-case corner, in accordance with traditional practice. However, the critical difference in nanometer technologies is the scale of these variations has shrunk, and even devices on the same die may behave very differently. This implies that some gates may become faster, and some slower, resulting in the possibility of cancellations of some variations. In this setting, using appropriate extreme-case process corners that ignore such cancellations will certainly produce functional circuits, but the guard-banding margin may be excessively conservative.

Why is this an important problem in practice? Excessive conservatism is likely to mean that circuit optimization will make most circuits much faster than they need to be—at the cost of increased power. Consequently, a chip may be unable to simultaneously meet the stringent set of delay and power specifications imposed upon it. On the other hand, insufficiently conservative margins may imply that a large number of manufactured chips will fail to meet specifications, resulting in yield losses that could sink a product, or even a company. Thus, getting it just right, and being able to predict the timing yield well, has a notable impact on the bottom line.

Enter the science of probability and statistical design. The rich history of this area certainly provides a springboard for solving the problem noted here. After all, since delays are a function of process parameters, which can be treated as random variables, statistical delay computation should be a simple matter of finding the distribution of a function of random variables. However, the problem is significantly more complex and involves numerous intricacies: any solution must scale to solve large-sized circuit blocks with millions of gates; correlations between variations, and between path delays, must be handled; simple closed-forms for these functions of random variables do not exist, even if the random variables are assumed to follow a standard distribution; altering the mind-set of a designer to think of a delay as a probability, rather than a deterministic number, is a major cultural change; and so on.

The chief idea of the paper is to use bounding techniques to capture the probability distribution of the circuit delay.

The challenge of developing fast and practical solutions has led to a substantial amount of research in this area the past few years, with elegant theory being applied to solve intensely practical problems.

The following paper by Orshansky and Wang is an excellent sample of the clever ideas being applied in this field. The chief idea of the paper is to use bounding techniques to capture the probability distribution of the circuit delay. This work leverages Slepian's inequality, a classical result in probability theory, that allows a correlated normally distributed function to be bounded, and brings in the idea of stochastic majorization, which enables a partial ordering to be established on stochastic inequalities. This approach is applied to find the cumulative distribution function (CDF) of the maximum delay of a set of paths, and applied to standard benchmark circuits.

The authors illustrate that on these circuits, the lower and upper bound are extremely close, and accurately match a Monte Carlo based evaluation of the CDF, at a fraction of the computational cost. In doing so, they clearly demonstrate how a set of classical results can be beautifully adapted and applied to solve a very practical engineering problem.

Sachin S. Sapatnekar (sachin@umn.edu) is a professor of Electrical and Computer Engineering at the University of Minnesota, Minneapolis, MN.

DOI:10.1145/1536616.1536641

Statistical Analysis of Circuit Timing Using Majorization

by Michael Orshansky and Wei-Shen Wang

ABSTRACT
Future miniaturization of silicon transistors following Moore's Law may be in jeopardy as it becomes harder to precisely define the behavior and shape of nanoscale transistors. One fundamental challenge is overcoming the variability in key integrated circuit parameters. In this paper, we discuss the development of electronic design automation tools that predict the impact of process variability on circuit behavior, with particular emphasis on verifying timing correctness. We present a new analytical technique for solving the central mathematical challenge of the statistical formulation of timing analysis which is the computation of the circuit delay distribution when delays of circuit elements are correlated. Our approach derives the bounds for the exact distribution using the theory of stochastic majorization. Across the benchmarks, the root-mean-square difference between the exact distribution and the bounds is 1.7–4.5% for the lower bound and 0.9–6.2% for the upper bound.

1. INTRODUCTION

A notable feature of silicon microelectronics at the nanometer scale is the increasing variability of key parameters affecting the performance of integrated circuits. This stems from the dramatic reduction in the size of transistors and on-chip wires, and is due to both technology-specific challenges and the fundamentals of nanoscale electronics. For example, one technology-specific challenge is the lack of a cost-effective replacement for using 193 nm light during the lithographic process, which yields poor fidelity when patterning transistors with a 20 nm length.

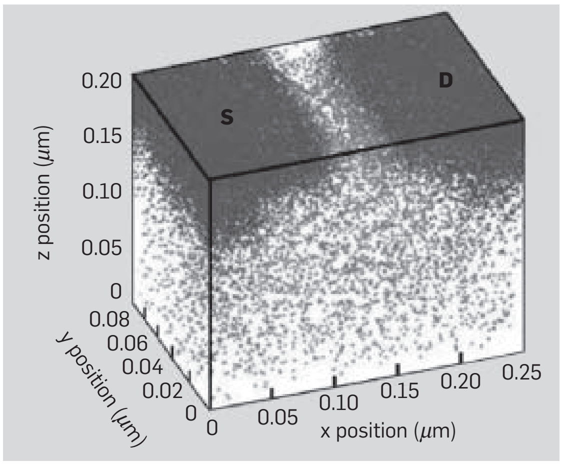
The more fundamental challenge is overcoming intrinsic randomness when manipulating materials on atomic scale. This randomness is, for example, manifested in threshold voltage variation, roughness of the transistor gate edge, and variation in the thickness of the dielectric transistor layer.[2] For example, the threshold voltage separating the on and off states of the transistor is controlled by adding dopant (non-silicon) atoms inside the transistor. Placing dopant atoms into a silicon crystal (see Figure 1) is essentially a random process, hence the number and location of atoms that end up in the channel of each transistor is likewise random. This leads to significant variation in the threshold voltage and many of the key electrical properties of the transistor.

Process parameters can be distinguished by the spatial scales on which they cause variability to appear, such as wafer-to-wafer, inter-chip, and intra-chip variability. Historically, intra-chip variability has been small but has grown due to the impact of technology-specific challenges in photolithography and wire fabrication, as well as the increased intrinsic randomness described above. Another important distinction is systematic versus random variability patterns. Systematic variation patterns are due to well-understood physical behaviors and thus can be modeled for a given chip layout and process. For example, the proximity of transistors to each other leads to a strong systematic effect on their gate lengths. Random variation sources, such as threshold voltage variation due to dopants, can only be described through stochastic modeling.

The variability in semiconductor process parameters translates to circuit-level uncertainty in timing performance and power consumption. The timing uncertainty is added to the uncertainty from the operating environment of circuit components, such as their temperature and supply voltage. The two sources of uncertainty, however, must be treated differently during the design process. Because a manufactured chip must operate properly under all operating conditions, environmental uncertainty is dealt with by using worst-case methods. The timing uncertainty due to process parameter variation can be dealt with statistically, since a small fraction of chips are allowed to fail their timing requirements and discarded during manufacturing.

Increased process variability presents challenges to IC design flows based on deterministic circuit analysis and

Figure 1. Distribution of dopant atoms in a transistor with the length of 50 nm. The number and location of atoms determine the key electrical properties. (Reprinted from Bernstein et al.[2] © IBM, 2006.)



A previous version of this article appeared in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2006).

optimization. This deficiency ultimately leads to a lower manufacturing yield and decreased circuit robustness, and results in designs that are sub-optimal in terms of silicon area or power consumption. A new design strategy is needed to seamlessly describe these variations and provide chip designers with accurate performance and robustness metrics.

2. BACKGROUND

Validating the correctness of timing behavior is a paramount task of design flow. For large digital systems, this is performed using static timing analysis (STA). In contrast to dynamic timing simulations which trace circuit response for specific inputs, static timing analysis ensures correct timing of synchronous digital circuits in an input-independent manner.[5]

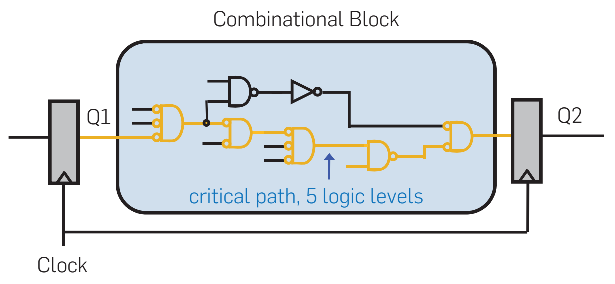
Virtually all current digital computing systems (such as microprocessors) are synchronous: all events are coordinated by a clock signal. A synchronous circuit is timing-correct if the signal computed by the combinational circuit is captured at the memory element, e.g., a flip-flop, on every latching edge of the clock signal (see Figure 2). The signal must arrive at the destination flip-flop by the time the latching clock edge arrives. The arrival time at the destination flip-flop is given by the path with the longest delay through the combinational circuit. Note that the clock signal arrival times vary from one memory element to another, and a complete timing analysis involves a simultaneous treatment of clock and data networks.

To carry out the timing checks, the circuit is represented by a graph with a vertex for each gate input/output and for the primary circuit inputs/outputs. A directed edge connects two vertices if the signal transition on the former causes a signal transition on the latter. Thus, the graph contains an edge for a transition between each possible input and output of the gate and for every interconnect segment. A virtual source node connected to all primary inputs is added to the graph, along with a virtual sink node connected to all primary outputs.

Definition: A timing graph $G = \{V, E, n_s, n_f\}$ is a directed graph having exactly one source node n_s and one sink node n_f, where V is a set of vertices and E is a set of edges. The weight associated with an edge corresponds to either the gate delay or interconnect delay.

An example of a simple circuit and its timing graph is shown in Figure 3. The timing graph is, or can be made, acyclic, thus the final data structure is a directed acyclic graph (DAG). The basic job of static timing analysis is to determine the maximum delay between the source and the sink node of the timing graph; this is the focus of the present article. STA is effective because it relies on algorithms with runtimes that grow only linearly with circuit size. For DAGs, computing the maximum propagation time can be performed using a graph traversal following a topological order completed in $O(|V| + |E|)$ time, where $|V|$ and $|E|$ denote the numbers of vertices and edges in the timing graph. In this method, the maximum arrival time to a given node is recorded and propagated, where the arrival time is the maximum time to propagate to a node through any path.

Figure 2. A combinational circuit with source and destination flip-flops.



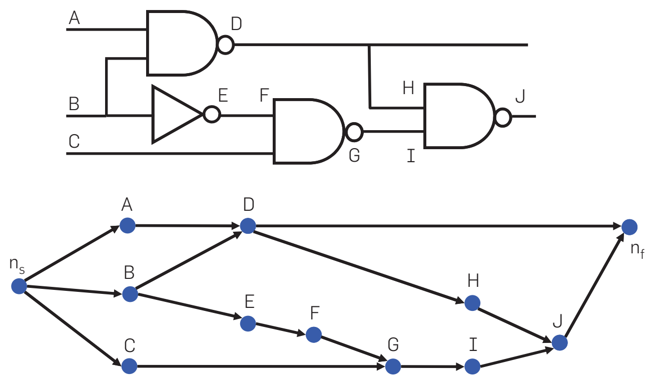
Traditional STA deals with the variability of process parameters deterministically by using the worst- and best-case corner strategy, in which edge delays are set to their maximum or minimum timing values that are possible as a result of process variation. This strategy fails for several reasons. First, corner-based deterministic STA produces increasingly conservative timing estimates, since it assumes that all edges behave in a correlated manner. This is reasonable only in the absence of intra-chip variability, which actually increases with scaling.[13] Furthermore, the probability of simultaneously having the process values corresponding to the corners shrinks with the growing dimensionality of the process space. Second, the computational cost of running a corner-based deterministic analysis becomes prohibitive since the number of corner cases to be checked grows exponentially with the number of varying process parameters. Third, because it is difficult to properly identify the corner conditions in the presence of intra-chip variation, deterministic timing analysis cannot always guarantee conservative results.

3. RELATED WORK

To overcome the limitations of deterministic timing, statistical static timing analysis (SSTA) has been proposed as an alternate way to accurately estimate circuit delay. SSTA models gate and interconnect delays as random rather than deterministic values and relies on fully probabilistic approaches to handle timing variability.

Definition: A timing graph G in which the ith edge weight d_i is random is called a probabilistic timing graph. A path p_i

Figure 3. (a) Gate schematic of a simple circuit and (b) its timing graph.



is a set of edges from the source node to the sink node in G. The delay of the path D_i is the sum of the weights d for all edges on the path. The goal of SSTA is to find the distribution of maximum $(D_1, \ldots, D_N)$ among all N paths in G.

Using a statistical treatment for STA requires an efficient method for processing probabilistic timing graphs with an arbitrary correlation of edge delays to find the distribution of maximum propagation time through the graph. This challenge has received significant attention by the electronic design automation (EDA) community.[1, 3, 8, 9, 17, 18]

Edge delay distribution depends on the distribution of the underlying semiconductor process parameters, which is typically assumed to be Gaussian. A linear dependence of edge delays on process parameters is also commonly accepted. Under this model, edge delays also have a Gaussian distribution. They are correlated because of the simultaneous effect of intra-chip and inter-chip variability patterns. While intra-chip variability impacts individual transistors and wires in an uncorrelated manner, inter-chip variability causes all edges of the timing graph to vary in a statistically correlated manner. Probabilistic timing graphs have been studied by the operations research community in the context of PERT (program evaluation and review technique) networks.[15] Earlier work, however, was largely concerned with the analysis of graphs with independent edge delay distributions. Even in this case, the exact distribution of graph propagation times cannot be generally obtained because arrival times become correlated due to shared propagation history.

The problem addressed by the EDA community is different in several respects: edge delays are correlated, estimating manufacturing yield requires capturing the entire cumulative distribution function (*cdf*) of duration time and not just the moments, and the network size is much larger (often with millions of nodes). Recent work in EDA has fallen into three categories: Monte-Carlo simulation,[6, 9] numerical integration methods in process parameters space,[8] and analytical techniques operating in edge delays space.[3, 17, 18] Monte-Carlo approaches estimate the distribution by repeatedly evaluating the deterministic STA algorithm, with delay values sampled from their distributions. These algorithms tend to be computationally expensive since thousands of runs may be needed to achieve reasonable accuracy. While being accurate, numerical integration methods have prohibitively high runtimes which limits them to only a very small number of independent process parameters.

It is not feasible to analytically compute the exact circuit delay distribution under correlated edge delays even under simple correlation structures.[12] This has led to solutions that either approximate[3, 17] or bound[1, 18] the distribution. Approximations are typically used in the so-called node-based methods that involve approximation of the node arrival time distributions. This allows for statistical timing evaluation in a way similar to the traditional topological longest path algorithm, in which each node and edge of a timing graph is explored in a topological order and is visited once in a single traversal of the graph.[3, 17] It thus preserves the linear runtime complexity of deterministic STA.

The computation of the node arrival time requires adding edge delays and then taking the maximum of edge delays. For two Gaussian variables, the sum is also Gaussian but the maximum $Z = \max(A, B)$ is not. Because node-based algorithms propagate arrival times through the timing graph, the edge delays and arrival times must be represented in an invariant manner when applying multiple add and max operators. The solution proposed in Jacobs et al.[7] is to approximate the result of the max computation at each node by a new Gaussian random variable. This is based on matching the first two moments of the exact distribution of Z with a new normal random variable. The method is efficient because the mean and variance of the maximum of two normal random variables can be computed in closed form.[4] In a timing graph with correlated edge delays, the new random variable must also reflect the degree of correlation it has with any other edge. For the known correlation between delay of edges A, B, and D, the correlation between $\max(A, B)$ and D can be computed in closed form.[4] Instead of storing the full correlation matrix directly, a linear model of delay as an explicit function of process parameters can be used to capture the correlation.[3]

If each edge in the graph adds some independent random variation to the total delay, then capturing arrival time correlation due to common path history requires storing information on every traversed edge. The can become overwhelming, however, because the number of edges may be quite large. A solution proposed in Chang and Sapatnekar[3] uses a dimensionality reduction technique of principal component analysis (PCA). If random components of delay contributed by timing edges belonging to the same region of the chip behave in a spatially correlated manner, then PCA is effective in reducing the number of terms to be propagated. An alternative strategy proposed in Visweswariah et al.[17] is to lump the contributions of all random variation into a single delay term and represent the node arrival times and edge delays as

$$a_o + \sum_{i=1}^{n} a_i \Delta P_i + a_{n+1} \Delta R_a,$$

where P_i is the ith inter-chip variation component, $\Delta P_i = P_i - P_o$ is its deviation from the mean value, a_i is the sensitivity of the gate or wire delay to the parameter P_i, ΔR_a is a standard normal variable, and the a_{n+1} coefficient represents the magnitude of the lumped random intra-chip component of delay variation. Representing random delay variation in this manner clearly cannot capture the correlation of arrival times due to signal paths sharing common ancestor edges. Yet keeping the path history makes the node-based algorithms less efficient, and known industrial applications[17] tend to ignore the impact of path reconvergence by relying on the simple delay model shown above.

For Gaussian process parameter variations, the linearized delay models described above permit efficient add and max operations and propagate the result in the same functional form. For the max operation, the result is mapped onto the canonical model using a linear approximation of the form $MAX(A, B) \approx C = T_A A + (1 - T_A)B$, where $T_A = P(A > B)$ is known as the tightness probability.

4. STATISTICAL STA USING BOUNDS

When computing the distribution of circuit delays, the quality of approximation degrades when applied to paths of equal criticality. In this case, the tightness probability is close to 0.5 and the linear approximation of the max operator becomes inaccurate, as illustrated below. As an example, in a perfectly balanced tree with 128 paths we have a node-based algorithm predicting a timing yield of 90% but the actual yield might be only 72% at some circuit delay values.

We propose a more reliable statistical estimation strategy using timing analysis that looks at a complete or large set of paths through the timing graph. The algorithm derives bounds on the circuit delay distribution using the distributions of path delays and their interdependencies.[18] While path tracing imposes an additional cost, it also makes it easier to handle more complex gate delay models, for example, to accurately model the delay dependency on the slope of the driving signal. Delay correlations introduced by path-sharing and joint impact of inter-chip and intra-chip variations are naturally taken into account, improving the overall accuracy.

The major difference between this algorithm and node-based methods is the focus on bounding rather than approximating the circuit distribution; this avoids the multiple approximations involved in node-based traversal. Experiments indicate that produced bounds are quite tight. We are specifically interested in the lower bound on the cdf of circuit delay since it provides a conservative value of the circuit delay at any confidence level. For example, across the benchmarks the error between the exact distribution and the lower bound is 1.1–3.3% (95th percentile).

The cost of this greater accuracy is longer runtimes. In the worst case, the number of paths in the circuit increases exponentially with the number of nodes. This can occur in arithmetic circuits, such as multipliers, but the number of paths for most practical circuits is actually quite manageable. A study of a class of large industrial circuits found the number of paths $\approx 0.12 \times \text{gates}^{1.42}$.[14] The availability of parallel multi-core systems makes storing and manipulating millions of paths practical. Recent industrial offerings of transistor-level static timing analysis, where the accuracy requirement is higher than for the gate-level analysis, have used path-based analysis and demonstrated fast high-capacity multithreaded implementations.[11]

Our algorithm separates the path extraction step from the statistical analysis of path delay distributions. It first extracts a set of the top N longest paths using deterministic STA. This is done using edge delays set to their mean values to extract paths with delays within a certain pre-defined range from the maximum delay. Path extraction can be done with time complexity of $O(N)$, which means only paths larger than a given threshold need to be traced.[19]

The result of the path extraction step is a set of N paths with their mean delay values (μ_{D_i}), variances ($\sigma^2_{D_i}$), and the path correlation matrix (Σ). The algorithm aims to find the distribution of the maximum delay of this set of paths. The complete description of a set of path delays is given by the cdf of D: $F(t) = P(\bigcap_{i=1}^{N}\{D_i \le t\})$, where D_i is the delay of the ith path in the circuit and $F(t)$ is the cumulative distribution function defined over the circuit delay probability space. The overall worst-case complexity of the algorithm is set by the path delay correlation matrix generation and is $O(N^2m^2)$, where N is the number of paths extracted and m is the maximum number of gates on the extracted paths.

The main contribution of our work is an algorithm that computes the upper and lower bounds on the probability distribution of circuit delays for a timing graph with correlated random edge delays. The algorithm uses a linear model of edge delays as a function of process parameters. We assume that process parameters are Gaussian, and under the above linear model, the edge delays are also Gaussian. The algorithm is based on a set of transformations that bound the circuit delay distribution by probabilities in the form of equicoordinate vectors with well-structured correlation matrices; these probabilities can be numerically precharacterized in an efficient manner.

4.1. Bounding the delay distribution

First, we express F(t) in terms of the distribution of a standard multivariate normal vector:

Theorem 1. For any normal random vector with a correlation matrix given by Σ

$$P_\Sigma(\bigcap\{D_i \le t\}) = P_\Sigma(\bigcap\{Z_i \le t'_i\})$$

where $t'_i = (t - \mu_{D_i}) / \sigma_{D_i}$ and $Z_i \sim N(0, 1)$ are the standard normal random variables.

The transformation introduces the vector t' that determines the set over which the probability content is being evaluated. Note that the components of the vector are not equal. Also note that the correlation matrix Σ that characterizes the path delay vector is populated arbitrarily and has no special structure. Both of these factors make the immediate numerical evaluation of the above probability impossible.

Second, we express the cumulative probability of Theorem 1 by a probability computed for a vector with a well-structured correlation matrix. We utilize a property unique to multivariate Gaussian distributions: their probabilities are monotonic with respect to the correlation matrix. Slepian has shown that by increasing the correlation between the members of the Gaussian vector, their probability volume over certain sets increases.[16] Formally:

Theorem 2. Let X be distributed as $N(0, \Sigma)$, where Σ is a correlation matrix. Let $R = (\rho_{ij})$ and $T = (\tau_{ij})$ be two positive semi-definite correlation matrices. If $\rho_{ij} \ge \tau_{ij}$ holds for all i, j, then for all $a = (a_1, \ldots, a_N)^T$, we have

$$P_{\Sigma=R}\left[\bigcap_{i=1}^{N}\{X_i \le a_i\}\right] \ge P_{\Sigma=T}\left[\bigcap_{i=1}^{N}\{X_i \le a_i\}\right]$$

It is easy to see that, as a special case, the above relation can be used to bound the sought probability with probabilities whose correlation matrices have identical off-diagonal components. Therefore, as a special case, the cumulative probability can be bounded from below and above by

$$P_\Sigma(\bigcap\{Z_i \le t'_i\}) \ge P_{\Sigma_{\min}}(\bigcap\{Z_i \le t'_i\})$$

$$P_{\Sigma}(\bigcap\{Z_i \le t_i'\}) \le P_{\Sigma_{\max}}(\bigcap\{Z_i \le t_i'\})$$

where $\Sigma_{\min}$ (and $\Sigma_{\max}$) are generated by setting all their off-diagonal elements to $\Sigma^{ij}_{\min} = \min\{\Sigma^{ij}\}$ (and $\Sigma^{ij}_{\max} = \max\{\Sigma^{ij}\}$) for all $i \neq j$. The bounding probabilities are thereby expressed in terms of simple and regular correlation matrices.

We now use majorization theory to evaluate the probability content of a standard multi-normal vector with a simple correlation structure over the non-equicoordinate set. For a more efficient numerical evaluation, we can resort to expressions in terms of the equicoordinate probability. This is attractive because it is easier to pre-characterize the probability of a multidimensional vector over a multidimensional semi-infinite cube rather than for all possible shapes of the multidimensional rectangular cuboid. We enable this transformation by bounding the cumulative probabilities of a normal random vector using the theory of stochastic majorization.[10] We use the property that for some distributions, including Gaussian, stochastic inequalities can be established based on partial ordering of their distribution parameter vectors using ordinary (deterministic) majorization.

We first introduce the notions of strong and weak majorization for real vectors. The components of a real vector $a = (a_1, \ldots, a_N)'$ can be arranged in increasing magnitude, $a_{[1]} \ge \ldots \ge a_{[N]}$. A majorization relationship is defined between two real vectors, a and b. We say that a majorizes b, in symbols $a \succ b$, if $\sum_{i=1}^{N} a_i = \sum_{i=1}^{N} b_i$ and $\sum_{i=1}^{r} a_{[i]} \ge \sum_{i=1}^{r} b_{[i]}$ for $r = 1, \ldots, N-1$. If only the second condition is satisfied, then only weak majorization holds. We say that a weakly majorizes b, in symbols $a \succ\succ b$, if $\sum_{i=1}^{r} a_{[i]} \ge \sum_{i=1}^{r} b_{[i]}$ for $r = 1, \ldots, N$. An example of strong majorization is $(3,2,1) \succ (2,2,2)$, and an example of weak majorization is $(3,2,1) \succ\succ (1,1,1)$.

We now relate ordering of parameter vectors to ordering of probabilities. Note that the class of random vectors that can be ordered via ordering of their parameter vectors includes Gaussian random vectors. Specifically:

Theorem 3. If $\underset{\sim}{t} = (t_1, \ldots, t_N)$ and $\underset{\sim}{\bar{t}} = (\bar{t}, \ldots, \bar{t})$, where $\bar{t} = \frac{1}{N}\sum_{i=1}^{N} t_i$, then

$$\underset{\sim}{t} \succ \underset{\sim}{\bar{t}} \rightarrow P(\bigcap\{Z_i \le t_i\}) \le P(\bigcap\{Z_i \le \bar{t}\})$$

which establishes an upper bound on the circuit delay probability. We use weak majorization for the lower bound. Specifically:

Theorem 4. If $\underset{\sim}{t} = (t_1, \ldots, t_N)$, $t_{\min} = \min(t_1, \ldots, t_N)$ and $\underset{\sim}{t}_{\min} = (t_{\min}, \ldots, t_{\min})$, then

$$\underset{\sim}{t} \succ\succ \underset{\sim}{t}_{\min} \rightarrow P(\bigcap\{Z_i \le t_{\min}\}) \le P(\bigcap\{Z_i \le t_i\})$$

Thus we have bounded the original probability by cumulative probabilities expressed in terms of the standard multivariate normal vector with an equicoordinate vector and a correlation matrix of identical off-diagonal elements:

$$P_{\Sigma_{\min}}(\bigcap\{Z_i \le t_{\min}\}) \le F(t) \le P_{\Sigma_{\max}}(\bigcap\{Z_i \le \bar{t}\})$$

This is a well-structured object whose probability content can be computed numerically. The numerical evaluation is done at pre-characterization time by Monte-Carlo integration of the cumulative distribution function of a multivariate normal vector. The result of pre-characterization is a set of probability lookup tables for a range of vector dimensionalities N, coordinate values t, and correlation coefficients.

We tested the algorithm on multiple benchmark circuits and compared our results to the exact cdf computed via Monte-Carlo runs of the deterministic STA algorithm. Samples were taken from the parameter distributions, and both inter-chip and intra-chip components were present.

Figure 4 shows the cumulative probabilities from the Monte-Carlo simulation and the derived bounds for one combinational benchmark circuit (c7552) containing 3874 nodes. The bounds are tight: across the

Figure 4. The derived bounds for cdf of delay of a benchmark circuit c7552 containing 3874 nodes.

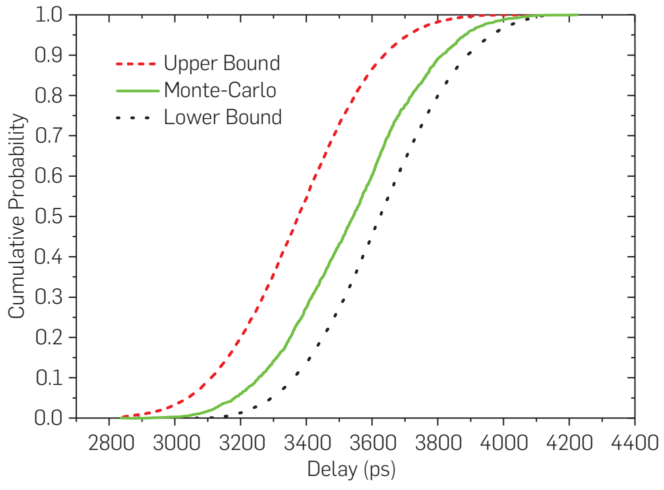


Figure 5. Change in the cdf for a benchmark circuit depending on the level of edge correlations.

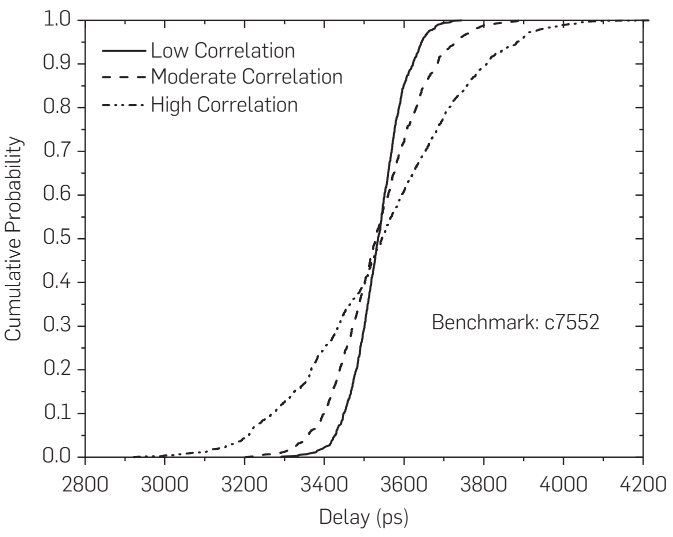
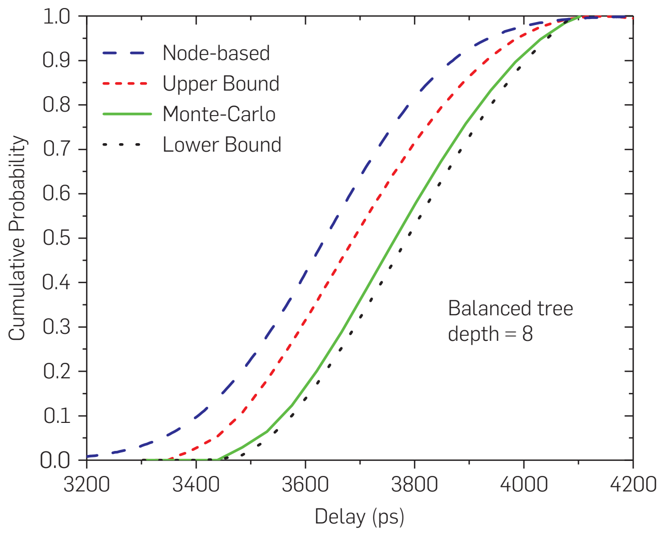


Figure 6. A comparison between the node-based algorithm and our bounding strategy for a balanced tree circuit.



benchmarks, the root-mean-square difference between the exact distribution and the bounds is 1.7–4.5% for the lower bound and 0.9–6.2% for the upper bound. Figure 5 illustrates that accurately accounting for edge delay correlations is crucial when predicting the shape of the cdf. The edge delay correlation changes with the ratio of the variance of inter-chip to intra-chip components. For example, the span of the low-correlation case is smaller than the high-correlation case. Our algorithm is more accurate than node-based approximation methods when processing balanced timing graphs with equal path mean delays and delay variances. In Figure 6, we compare the cumulative distribution functions generated by the node-based algorithm using the procedure of Visweswariah et al.[17] against the proposed approach. The cdfs are generated for a balanced tree circuit with depth of 8. An equal breakdown of total variation into inter-chip and intra-chip variability terms was used. Note that the approximate cdf may be noticeably different from the true one, but the bounds accurately contain the true cdf.

The implementation is very efficient even though the algorithm runtime is quadratic in the number of deterministically longest paths. The C++ implementation ran on a single-core machine with a 3.0 GHz CPU and 1 GB memory and took less than 4 seconds for the largest circuit in the ISCAS '85 benchmark suite (3874 nodes). When evaluated at a fixed number of extracted paths, there is a close-to-linear growth in algorithm runtime as a function of circuit size.

5. CONCLUSION

We have presented a new statistical timing analysis algorithm for digital integrated circuits. Instead of approximating the cdf of a circuit, we use majorization theory to compute a tight bound for the delay cdf. The equicoordinate random vectors are used to bound the exact cumulative distribution function, which facilitates the numerical evaluation of the probability.

Acknowledgments

We thank Ashish K. Singh for his help with the simulations, and Sachin Sapatnekar for his feedback on the article. The work has been partially supported by the NSF Career grant CCF-0347403.

References

1. Agarwal, A., Zolotov, V., Blaauw, D.T. Statistical timing analysis using bounds and selective enumeration. *IEEE Trans. Comp. Aided Des. Integr. Circuits Syst. 22*, 9 (2003), 1243–1260.
2. Bernstein, K., Frank, D.J., Gattiker, A.E., Haensch, W., Ji, B.L., Nassif, S.R., Nowak, E.J., Pearson, D.J., Rohrer, N.J. High-performance CMOS variability in the 65-nm regime and beyond. *IBM J. Res. Dev. 50*, 4 (2006), 433–449.
3. Chang, H., Sapatnekar, S.S. Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In *Proceedings of the International Conference on Computer-Aided Design* (San Jose, CA, 2003).
4. Clark, C.E. The greatest of a finite set of random variables. *Oper. Res. 9*, 2 (1961), 85–91.
5. Hassoun, S., Sasao, T. *Logic Synthesis and Verification*. Kluwer Academic Publishers, New York, 2002.
6. Hitchcock, R.B. Timing verification and the timing analysis program. In *Proceedings of the 19th Conference on Design Automation* (Piscataway, NJ, 1982).
7. Jacobs, E.T.A.F., Berkelaar, M.R.C.M. Gate sizing using a statistical delay model. In *Proceedings of the Conference on Design, Automation and Test in Europe* (Paris, France, 2000).
8. Jess, J.A.G., Kalafala, K., Naidu, S.R., Otten, R.H.J.M., Visweswariah, C. Statistical timing for parametric yield prediction of digital integrated circuits. In *Proceedings of the 40th Conference on Design Automation* (Anaheim, CA, 2003).
9. Jyu, H.F., Malik, S., Devadas, S., Keutzer, K.W. Statistical timing analysis of combinational logic circuits. *IEEE Trans. VLSI Syst. 1*, 2 (1993), 126–137.
10. Marshall, A.W., Olkin, I. *Inequalities: Theory of Majorization and Its Applications*. Academic Press, New York, 1979.
11. Moretti G. CLK Design Automation introduces Amber FX Analyzer, *EDA Design Line*. Retrieved on May 22, 2009, http://www.edadesignline.com/.
12. Nadas, A. Probabilistic PERT. *IBM J. Res. Dev. 23*, 3 (1979), 339–347.
13. Orshansky, M., Nassif, S., Boning, D. *Design for Manufacturability and Statistical Design: A Constructive Approach*. Springer, New York, 2008.
14. Ramalingam, A., Nam, G., Singh, A.K., Orshansky, M., Nassif, S.R., Pan, D.Z. An accurate sparse matrix based framework for statistical static timing analysis. In *Proceedings of the International Conference on Computer-Aided Design* (San Jose, California, 2006).
15. Robillard, P., Trahan M. The completion time of PERT networks. *Oper. Res. 25*, 1 (1977), 15–29.
16. Tong, Y.L. *Probability Inequalities in Multivariate Distributions*. Academic Press, New York, 1980.
17. Visweswariah, C., Ravindran, K., Kalafala, K., Walker, S.G., Narayan, S. First-order incremental block-based statistical timing analysis. In *Proceedings of the 41st Conference on Design Automation* (San Diego, CA, 2004).
18. Wang, W.S., Orshansky, M. Path-based statistical timing analysis handling arbitrary delay correlations: theory and implementation. *IEEE Trans. Comp. Aided Des. Integr. Circuits Syst. 25*, 12 (2006), 2976–2988.
19. Yen, S.H., Du, D.H., Ghanta, S. Efficient algorithms for extracting the K most critical paths in timing analysis. In *Proceedings of the 26th Conference on Design Automation* (Las Vegas, NV, 1989).

Michael Orshansky (orshansky@cerc.utexas.edu), Department of ECE, University of Texas, Austin.

Wei-Shen Wang (wei-shen wang@mentor.com), Mentor Graphics, Taipei, Taiwan.

Denison University
Assistant Professor of Computer Science

Denison University invites applications for a tenure track position in Computer Science, to begin in January or August 2010. Candidates must have earned a Ph.D. in Computer Science or a closely related field. We are seeking an energetic individual who is committed to teaching a variety of CS courses to undergraduates in a liberal arts setting, supervising undergraduate research, and maintaining a strong scientific research program. The current CS teaching load is 9 courses every 2 years.

Denison University is a highly selective, private liberal arts college enrolling approximately 2,100 undergraduate students. Denison is located in Granville, Ohio, 25 miles east of Columbus. For more information, please see our website at http://www.denison.edu/academics/departments/mathcs/

The Department of Mathematics and Computer Science offers B.A. and B.S. degrees in both computer science and mathematics. Our department is home to 4 computer scientists and 6 mathematicians; we consider collaboration between members of the two fields to be one of our strongest assets.

To apply, please submit a letter of application, a curriculum vita, graduate transcripts, statements on your teaching philosophy and research program, and three letters of recommendation online at https://employment.denison.edu

At least one recommendation letter must address your teaching effectiveness or potential.

We will begin reviewing applications on September 15, 2009 and will continue until the position is filled. Denison University is an Affirmative Action/Equal Opportunity Employer. Women and minority candidates are especially encouraged to apply.

The Hong Kong Polytechnic University
Department of Computing

The Department invites applications for Professors/Associate Professors/Assistant Professors in Database and Information Systems / Biometrics, Computer Graphics and Multimedia / Software Engineering and Systems / Networking, Parallel and Distributed Systems. Applicants should have a PhD degree in Computing or closely related fields, a strong commitment to excellence in teaching and research as well as a good research publication record. Applicants with extensive experience and a high level of achievement may be considered for the post of Professor/Associate Professor. Please visit the website at http://www.comp.polyu.ed.hk for more information about the Department. Salary offered will be commensurate with qualifications and experience. Initial appointments will be made on a fixed-term gratuity-bearing contract. Re-engagement thereafter is subject to mutual agreement. Remuneration package will be highly competitive. Applicants should state their current and expected salary in the application. Please submit your application via email to hrstaff@polyu.edu.hk. Application forms can be downloaded from http://www.polyu.edu.hk/hro/job.htm. **Recruitment will continue until the positions are filled.** Details of the University's Personal Information Collection Statement for recruitment can be found at http://www.polyu.edu.hk/hro/jobpics.htm.

University of Cape Town
Senior Lecturer

The Department of Computer Science seeks to appoint an academic staff member at Senior Lecturer level.

Technical University of Denmark DTU

ASSOCIATE/ASSISTANT PROFESSOR

Language Based Technology

DTU Informatics invites applications for one or more open positions as Associate or Assistant Professor in the section on Language Based Technology. The positions are available from 1 January 2010. The department covers informatics and mathematical modelling including language based technology, that is, techniques and tools for modelling, analysing and realising systems based on formal techniques – including static analysis and model checking.

Application deadline: **7 September 2009**

DTU is a leading technical university rooted in Denmark but international in scope and standard. Our total staff of 4,500 is dedicated to create value and to promote welfare for the benefit of society through science and technology; and our 6,000 students are being trained to address the technological challenges of the future. While safeguarding academic freedom and scientific independence we collaborate with business, industry, government, public agencies as well as other universities around the world.

Further details: dtu.dk/vacancy

acm

ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: Send an e-mail to *acmmediasales@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 by typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.

Rates: $325.00 for six lines of text, 40 characters per line. $32.50 for each additional line after the first six. 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 30 days.
For More Information Contact:
ACM Media Sales
at 212-626-0686 or
*acmmediasales@acm.org***

The successful candidate must have a PhD prior to taking up the appointment and will be expected to develop and teach courses at all levels, to carry out research, and to supervise postgraduate students in his/her area of specialisation. We particularly seek applications from candidates with research and teaching interests in Artificial Intelligence, but a candidate whose research field is compatible with that of any research group in the department is also encouraged to apply.

The Department offers a wide variety of courses and has a substantial cohort of MSc and PhD students. Our Honours degrees are accredited by the British Computer Society.

The Department hosts the UCT Centre in Information and Communication Technology for Development, and also specialises in telecommunications, visual computing, high-performance computing, digital libraries and information management. Further information on our research groups and activities can be found at http://www.cs.uct.ac.za

The annual remuneration package for a Senior Lecturer, including benefits, is R433 117.

Please send/email: a letter of application, your CV (no certificates), a one-page summary of your CV, and email and telephone details of 3 contactable referees to: Mr. Themba Mabambi (Ref: 2017), Staff Recruitment & Selection, UCT, Rondebosch 7701, Cape Town by **24 August 2009**.

Email: Themba.Mabambi@uct.ac.za,

Tel: (021) 650-2220

UCT is committed to the pursuit of excellence, diversity and redress. Our Employment Equity Policy is available at http://hr.uct.ac.za/policies/ee.php.

University of Washington

Department of Computer Science & Engineering

Research Assistant Professor

The University of Washington's Department of Computer Science & Engineering has one or more open Research Faculty positions in the area of computer vision, with an emphasis on 3D Modeling. The appointment will be a non-tenured research position at the rank of Research Assistant Professor. Applicants must have earned a doctorate by the date of appointment. All faculty engage in teaching, research, and service.

Please apply online at
http://www.cs.washington.edu/news/jobs.html"
http://www.cs.washington.edu/news/jobs.html

with a letter of application, a complete curriculum vitae, statement of research interests, and the names of four references. Applications received by September 1, 2009 will be given priority consideration.

Open positions are contingent on funding.

The University of Washington was awarded an Alfred P. Sloan Award for Faculty Career Flexibility in 2006. In addition, the University of Washington is a recipient of a National Science Foundation ADVANCE Institutional Transformation Award to increase the participation of women in academic science and engineering careers. We are building a culturally diverse faculty and encourage applications from women and minority candidates. The University of Washington is an affirmative action, equal opportunity employer.

Windows

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.

Call for Nominations

The ACM Doctoral Dissertation Competition

RULES OF THE COMPETITION
ACM established the Doctoral Dissertation Award program to recognize and encourage superior research and writing by doctoral candidates in computer science and engineering. These awards are presented annually at the ACM Awards Banquet.

SUBMISSIONS: Nominations are limited to one per university or college, from any country, unless more than 10 Ph.D.'s are granted in one year, in which case two may be nominated.

DEADLINE: Submissions must be received at ACM headquarters by October 30, 2009 to qualify for consideration.

ELIGIBILITY: Each nominated dissertation must have been accepted by the department between October 2008 and September 2009. Only English language versions will be accepted. Please send a copy of the thesis in PDF format to emily.eng@acm.org.

SPONSORSHIP: Each nomination shall be forwarded by the thesis advisor and must include the endorsement of the department head. A one-page summary of the significance of the dissertation written by the advisor must accompany the transmittal.

PUBLICATION RIGHTS: Each nomination must be accompanied by an assignment to ACM by the author of exclusive publication rights.

PUBLICATION: Winning dissertations will be published by Springer.

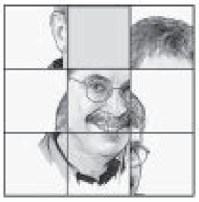
SELECTION PROCEDURE: Each nominated dissertation will be screened by three to six specialists in the relevant fields to provide a preliminary evaluation.

On the basis of these evaluations, five finalists will be chosen for submission to a selection committee of six persons who serve staggered terms of three years.

The selection committee will select the winning dissertation in early 2010.

AWARD: The Doctoral Dissertation Award is accompanied by a prize of $20,000 and the Honorable Mention Award is accompanied by a prize of $10,000. Financial sponsorship of the award is provided by Google.

http://awards.acm.org/html/dda.cfm



DOI:10.1145/1536616.1536642

Peter Winkler

Puzzled
Probability and Intuition

Welcome to three new puzzles. Solutions to the first two will be published next month; the third is (as yet) unsolved. In each puzzle, the issue is how your intuition matches up with the mathematics.

1. It is your last night in Las Vegas as you celebrate your 29th birthday. Standing at the roulette table with \$105 in your pocket, you resolve to make 105 successive \$1 bets on the number 29. You will win \$36 (minus your \$1 bet) each time the ball lands on "29," but, unfortunately, this happens with probability only 1/38; the rest of the time you simply lose your dollar. Use your intuition. What is the probability that, after the 105 bets, you come out ahead?

2. A hundred people board a fully booked aircraft. Unfortunately, the first person in line somehow loses his/her boarding pass while entering and takes a random seat. Each successive passenger then sits in his/her proper seat, if available; otherwise, each one rather wimpily takes a random vacant seat. Again, use your intuition. What is the probability that the last passenger finds the properly assigned seat unoccupied?

3. The Random Arcade, a favorite hangout of local video gamers, boasts a line of n gumball machines. Each machine is unpredictable but produces an average of one gumball each time it is operated; for example, it may be that Machine no. 1 produces two balls half the time and the rest of the time none at all.

What is the maximum possible probability that if you put a coin in each machine, you will be rewarded with a total of more than n gumballs?

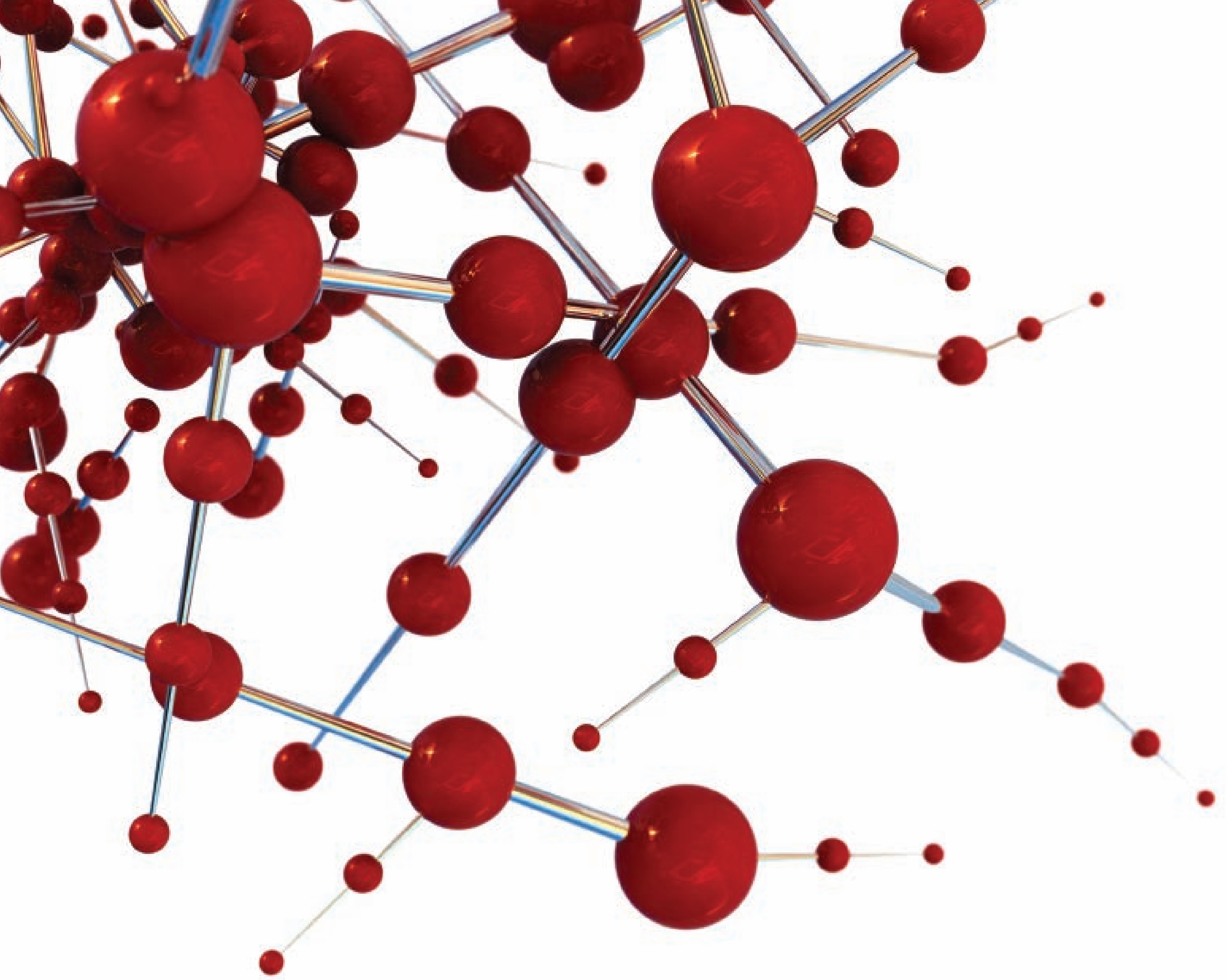
This puzzle (stated in terms of sequences of independent random variables) is due to Uriel Feige of the Weizmann Institute of Science, Rehovot, Israel. My intuition, and perhaps yours, too, suggests that the best possible situation is if each gumball machine disgorges $n+1$ gumballs with probability $1/(n+1)$, otherwise it gives nothing. That way, you succeed as long as at least one of the n machines pays off. What is your success probability?

Failure requires that every machine refuses to cooperate, happening with probability $(1 - 1/(n+1))^n$. So you succeed with probability one minus that expression. For $n = 1$ through 6, this gives success probabilities of 1/2, 5/9, 37/64, 369/625, 4,651/7,776, and 70,993/117,649; to the nearest thousandth, these numbers are .500, .556, .578, .590, .598, and .603. The numbers approach $1 - 1/e \sim .632$ from below as n increases. Thus, the answer appears to be $1 - 1/e$.

However, despite some serious effort, no one has managed to prove that you can't do better than $1 - 1/e$. Feige himself showed that the success probability can never exceed $12/13 \sim .923$. Can you improve on his bound?

Readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is Professor of Mathematics and of Computer Science and Albert Bradley Third Century Professor in the Sciences at Dartmouth College, Hanover, NH.



Think Parallel.....

It's not just what we make.
It's what we make possible.

Advancing Technology Curriculum
Driving Software Evolution
Fostering Tomorrow's Innovators

Learn more at: www.intel.com/thinkparallel