COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

06/2013 VOL.56 NO.6

Shafi Goldwasser
Silvio Micali

ACM's A.M. Turing Award Recipients

Association for
Computing Machinery

15th International Conference on Human-Computer Interaction with Mobile Devices and Services

MOBILEHCI 2013

Munich, Germany

August 27-30

@ http://www.mobilehci2013.org

http://www.facebook.com/MobileHCI2013

@MobileHCI2013

LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

acm

COMMUNICATIONS OF THE ACM

Departments

Last Byte

Association for Computing Machinery
Advancing Computing as a Science & Profession

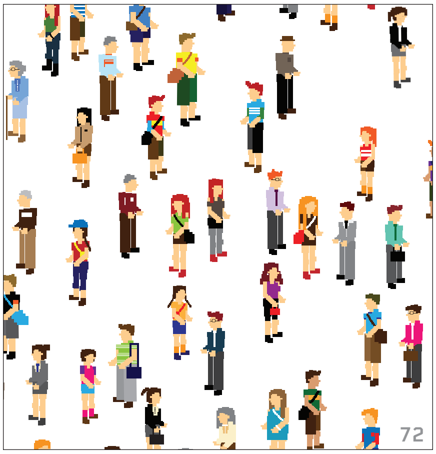
News

Viewpoints

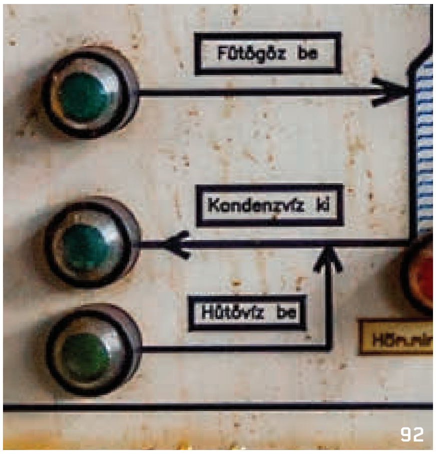
Practice

Articles' development led by acmqueue
queue.acm.org

Contributed Articles



Review Articles



Research Highlights

IMAGE BY SYED REZA ALI; PIXEL PEOPLE FROM SHUTTERSTOCK.COM; PHOTOGRAPH BY OLIVER SYED

About the Cover:
Two covers, two sides of the 2012 ACM A.M. Turing recipients Shafi Goldwasser and Silvio Micali, honored for their transformative work that laid the foundation for the science of cryptography, which resulted in pioneering new methods for efficient verification of mathematical proofs in complexity theory. Whichever cover the reader receives is a nod to the 'randomness' that played a great role in their work.
Photographer: Bryce Vickmark

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

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

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

Executive Director and CEO
John White
Deputy Executive Director and COO
Patricia Ryan
Director, Office of Information Systems
Wayne Graves
Director, Office of Financial Services
Russell Harris
Director, Office of SIG Services
Donna Cappo
Director, Office of Publications
Bernard Rous
Director, Office of Group Publishing
Scott E. Delman

ACM COUNCIL
President
Vinton G. Cerf
Vice-President
Alexander L. Wolf
Secretary/Treasurer
Vicki L. Hanson
Past President
Alain Chesnais
Chair, SGB Board
Erik Altman
Co-Chairs, Publications Board
Ronald Boisvert and Jack Davidson
Members-at-Large
Eric Allman; Ricardo Baeza-Yates; Radia Perlman; Mary Lou Soffa; Eugene Spafford
SGB Council Representatives
Brent Hailpern; Joseph Konstan; Andrew Sears

BOARD CHAIRS
Education Board
Andrew McGettrick
Practitioners Board
Stephen Bourne

REGIONAL COUNCIL CHAIRS
ACM Europe Council
Fabrizio Gagliardi
ACM India Council
Anand S. Deshpande, PJ Narayanan
ACM China Council
Jiaguang Sun

PUBLICATIONS BOARD
Co-Chairs
Ronald F. Boisvert; Jack Davidson
Board Members
Marie-Paule Cani; Nikil Dutt; Carol Hutchins; Joseph A. Konstan; Ee-Peng Lim; Catherine McGeoch; M. Tamer Ozsu; Vincent Shen; Mary Lou Soffa

ACM U.S. Public Policy Office
Cameron Wilson, Director
1828 L Street, N.W., Suite 800
Washington, DC 20036 USA
T (202) 659-9711; F (202) 667-1066

Computer Science Teachers Association
Chris Stephenson,
Executive Director

STAFF

DIRECTOR OF GROUP PUBLISHING
Scott E. Delman
publisher@cacm.acm.org

Executive Editor
Diane Crawford
Managing Editor
Thomas E. Lambert
Senior Editor
Andrew Rosenbloom
Senior Editor/News
Larry Fisher
Web Editor
David Roman
Editorial Assistant
Zarina Strakhan
Rights and Permissions
Deborah Cotton

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Directors
Mia Angelica Balaquiot
Brian Greenberg
Production Manager
Lynn D'Addesio
Director of Media Sales
Jennifer Ruzicka
Public Relations Coordinator
Virginia Gold
Publications Assistant
Emily Williams

Columnists
Alok Aggarwal; Phillip G. Armour; Martin Campbell-Kelly; Michael Cusumano; Peter J. Denning; Shane Greenstein; Mark Guzdial; Peter Harsha; Leah Hoffmann; Mari Sako; Pamela Samuelson; Gene Spafford; Cameron Wilson

CONTACT POINTS
Copyright permission
permissions@cacm.acm.org
Calendar items
calendar@cacm.acm.org
Change of address
acmhelp@acm.org
Letters to the Editor
letters@cacm.acm.org

WEBSITE
http://cacm.acm.org

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

ACM ADVERTISING DEPARTMENT
2 Penn Plaza, Suite 701, New York, NY 10121-0701
T (212) 626-0686
F (212) 869-0481

Director of Media Sales
Jennifer Ruzicka
jen.ruzicka@hq.acm.org

Media Kit acmmediasales@acm.org

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

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

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

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

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

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

Association for Computing Machinery

Printed in the U.S.A.

PLEASE RECYCLE THIS MAGAZINE

DOI:10.1145/2461256.2461257

Erik R. Altman

SGB Fortifies Global SIG Community

ACM's Special Interest Groups (SIGs) cover 36 areas of computing, from graphics to computer-human interfaces, theory to computer architecture, programming languages to bioinformatics, and much more. In fact, SIGs form the locus of ACM's technical activities and provide a way for individual disciplines to organize their interests. Each SIG holds an annual business meeting, where members learn more about the group's activities, propose new activities, and comment about SIG and ACM policies.

The SIG Governing Board (SGB), made up of representatives from all ACM SIGs, serves as the venue in which these ideas, concerns, and proposals can be shared across disciplines. The major role of the board is to strengthen and facilitate communication between SIGs and to work as one unit to build strategies and create opportunities for ACM's SIGs to thrive. The SGB is also charged with ensuring all SIGs are run well, forming new SIGs, discussing and implementing changes to broad policies, and providing technical advice and grass-roots input from the broad set of communities to ACM as a whole.

The board meets twice a year and a major component of these gatherings is to discuss best practices among SIGs. These discussions in turn allow for key community ideas to be circulated and adopted by a broad set of SIGs or ACM as a whole. For example, there has been much debate in recent years about ACM's publishing model. Comments from SIG members about Open Access have been widely discussed in SGB meetings and elsewhere in ACM, and have helped shape ACM's new Fair Access policy, which is described by ACM's Publications Board chairs Ronald Boisvert and Jack Davidson in the February 2013 issue of *Communications* (p. 5; http://bit.ly/12clIRS).

New SIGs arise regularly as new disciplines and groupings emerge. SIGBio (bioinformatics) and SIGHPC (High Performance Computing) are examples of two new additions to the SIG roster. Alas some SIGs lose relevance, become outdated, entwined with other disciplines, or just dissolve. It's all part of the SIG evolution.

SIGs serve as a community forum. A common thread running through all ACM SIGs is conferences. SIGs host and/or sponsor conferences within their disciplines throughout the year. These conferences are offered at member discounts and provide organizational continuity. SIGs also provide financial support when conferences incur a deficit. "Conference sponsorship" is often confusing. Companies and other organizations frequently donate funds to help defray conference costs—this sponsorship is purely voluntary, ad hoc, and changes yearly. ACM SIGs, however, provide *legal* sponsorship, for example, guaranteeing conference liabilities and ownership of the conference name, which requires formal legal agreement to change. Hence, SIG-sponsored conferences tend to be stable over long periods of time.

SIGs serve as a focal point: bringing together top research as well as changing activities to remain relevant and reflective of community needs.

Member benefits may vary by SIG, however, most SIGs support a number of other activities including newsletters, journals, blogs, wikis, awards recognizing excellence, and travel grants—especially for students and others who may face special hardship.

With all of their activities and shaping of policy, SIGs serve as a focal point: bringing together top research as well as changing activities to remain relevant and reflective of community needs. I encourage everyone to become a SIG member and to consider SIG volunteer roles—helping with conferences, newsletters, registration, or whatever may be appropriate for SIGs of interest to you. Such participation helps ensure your voice is heard and helps create better offerings for the community. C

Erik R. Altman is manager of the Dynamic Optimization Group at IBM's T.J. Watson Research Center, Yorktown Heights, NY, and chair of ACM's SIG Governing Board.

DOI:10.1145/2461256.2461258

Vinton G. Cerf

Honoring Our Best

It is June again and we gather once more in San Francisco to honor the best among us. It is proper, important, and fitting that we should do this. Not only to celebrate the successes and contributions of our colleagues but to convey to the general public the remarkable power of computer science and the value its practitioners and theorists give to the world. In many cases, the value is easily expressed and understood by many who benefit, but in many others, the work is more obscure and its utility less obvious and may be realized only long after its initial achievement. We do our colleagues and the general public a service when we explain and honor the work of these men and women and remind them of the increasing importance of computing and the myriad applications this discipline has spawned.

We owe the ACM Awards Committee and the many members who lead and serve on it our gratitude. It is a difficult task to evaluate the candidates and their work and to place it in the constellation of other work that has preceded it. Of course, none of this can be accomplished without nominations to work from and that is where you come in. Anyone who has ever prepared a nomination or endorsed one will know it is not an easy task, but it is a vital one. The universe of computer science has expanded dramatically since its earliest period and it is impossible to be expert in all things. The members of the awards committee are helped enormously in their task by well written, thoughtful, and, above all, clear nominations that explain the value, impact, and importance of the work to be recognized. Placing it in the perspective of earlier work, articulating the advances it has made, explaining its utility where this is not apparent are all extremely helpful.

In some cases, more than one person has contributed to the work to be honored and it is appropriate to recognize a small group. One of the most difficult tasks of the nominators and the award committee members is to assess whether the work to be recognized is properly represented by the awardees.

A new non-ACM award, The Queen Elizabeth II Prize for Engineering, illustrates this point. The prize of one million pounds Sterling is offered to a group of up to three honorees. It is a major new award to be made biannually and it complements the Nobel prizes in Science and the ACM A.M. Turing Award that is specifically for contributions to computer science. It is important because it recognizes that engineering has as profound an influence as science since its roots are found in the application of science to improve the quality of our lives.

We do colleagues and the public a service when we honor the work of these men and women and remind them of the importance of computing and the myriad applications it has spawned.

The first prize granted will be presented by Her Majesty, Queen Elizabeth II, in June 2013 not long after the ACM Awards banquet on June 15. The award committee received a nomination for a group achievement: The Internet and the World Wide Web. You may imagine what a challenge it was for the committee to attempt to select no more than three people to be recognized for this engineering feat. In fact, they could not and persuaded the foundation leadership that funds the award to permit them to recognize five individuals. The award rules were changed by the foundation leadership, more or less at the last minute, to accommodate this request. We have had similar experiences in the ACM Awards process, particularly in the Software System Award and Turing Award where three individuals (or more, in the case of the Software System Award) have occasionally been recognized for their extraordinary and collaborative work.

Returning to the principal theme of this column, I invite you to review the ACM Awards pages (http://awards.acm.org/html/awards.cfm) and to give serious thought to nominating colleagues whose work meets the award criteria and should be recognized. It is vital work because it helps ACM highlight and honor the work of computer scientists and engineers and it helps us to explain the value of that work to the general public.

Vinton G. Cerf, ACM PRESIDENT

DOI:10.1145/2461256.2461260

How to Claim Your Fair Share in Academic Publishing

IN HIS EDITOR'S LETTER "To Boycott or Not to Boycott" (Mar. 2013), Moshe Y. Vardi said the traditional author partnership with commercial publishers has turned into an abusive relationship. It is time computer scientists broke off that relationship, though not by boycotting, politicking, moralizing, or shedding tears, but rather by asking a fair price for the work we do as authors and as reviewers. With every niche activity organized as a profit center today, core contributions like authoring and reviewing should no longer be provided for free to for-profit publishers. The fees we ask for our contribution to publishing should increase with the price we pay for the respective journal. Moreover, the fees should go to the authors' institutions' libraries, not to the authors and reviewers directly. We must collectively think in terms of market forces, not giveaways.

Andreas Siebert, Landshut, Germany

Moshe Y. Vardi said, "I believe in keeping science separate from politics" (Mar. 2013) but may well have proclaimed, long after Napoleonic infantry tactics became impractical, "I believe we must fire only from the field, in square or in line."

Is not the form of battle determined by the nature of the obstacles, as well as the nature of those who carry it forward and of the opponent's objectives?

That we may pressure them to reform their business practices, Vardi urged us to confront publishers "the old-fashioned way, by out-publishing them." I agree. Out-publish them... with robust vitality.

However, when attacked indirectly, asymmetrically, as when publishers lobby the sources of research funds, CS authors and editors alike must respond in kind. Two elements of such a response are those Vardi found disquieting: a boycott and influencing one's peers. If among their opponents are colluding corporatized research publishers, then authors must expect to have their actions, along with their strength and resolve, tested on many fronts.

Editors of influential publications cannot waiver at first fire. Rather than ask, "Where shall we draw the line?," we should expect thoughtful, effective, farsighted leadership.

If authors wish the battle to be returned (and remain with) content, their response must be quick and unified. However, they might also find publishers are not the sole impediment and so must also weigh their readers and their products.

CS authors' first few steps toward discovery are short, starting with a look in a mirror to give themselves understanding by first donning their institutional regalia.

Nick Ragouzis, San Francisco, CA

What an Algorithm Is, and Is Not

Addressing the question "What Is an Algorithm?" in his Editor's Letter (Mar. 2012), Moshe Y. Vardi wrote that there is no consensus, suggesting the two abstract approaches of Yuri Gurevich[1] and Yiannis N. Moscovachis[2] were both correct; algorithms are the abstract state machines of Gurevich and the recursors of Moscovachis. This is (as Moscovachis said) like trying to explain that the number 2 is the set {Ø, {Ø}}, which is likely to leave an outsider less, not more, enlightened.

A proper definition would allow computer scientists to talk to the public about algorithms. The pragmatic motivation is to justify the treatment we informally apply to algorithms: construct, explain, and exchange; admire for elegance; debate whether one is the same as another; marvel that the one for the program Eliza is so simple; patent or sell them; or preclude protections based on principled argument. The theoretical motivation is to probe relationships to programs and abstract machines and promote discussion of semantics and implementation.

Now consider this definition: An algorithm is an abstract deterministic control structure accomplishing a given task under given conditions, expressed in a finite, imperative form. A critical look would reveal no formalities but plenty to consider, along with some surprises.

The definition includes compass-and-straight-edge algorithms (such as to bisect an angle) and other nonelectronic but straightforward sets of instructions. Excluded are recipes, which are not deterministic except in perversely rigorous cases. Also excluded are games, which, construed as a single agent's instructions, are not imperative control structures that accomplish a given task (with certainty) and which, construed as interchanges of moves, are not deterministic.

Perhaps most controversial, it also excludes recursive definitions, which are not imperative but declarative. A definition of the factorial function with a base case and a recursive case does not undertake a computation of a factorial or provide directions to do so. An algorithm must "do" not "be." (Gurevich noted that one possible algorithm generally available through such a definition starts with "Apply the equations..."; that is, the algorithm tells us the obvious thing to do with the definition.) The imperative requirement yields the interesting result that computers, at the hardware level, do not execute algorithms. Current flowing through circuits does not express an imperative. Algorithms are a human construct.

So, do all programs implement algorithms? People are the interpreters of the elements of the definition, responsible for measuring the concept against them. But how do we map the path from the human view to the mathematical view? Gurevich and

Moscovachis each suggested the rich connotations of the concept of the algorithm cannot be captured fully through a recurrence relation or other set-theoretic object. We can honor their contributions by pursuing the questions raised by the definition I have outlined here or through a competing informal view.

Robin K. Hill, Laramie, WY

References

1. Gurevich, Y. *What is an algorithm?* Technical Report. Microsoft Research, Redmond, WA, 2011; http://research.microsoft.com/en-us/um/people/gurevich/Opera/209.pdf
2. Moscovachis, Y.N. *On founding the theory of algorithms.* UCLA Department of Mathematics, Los Angeles, CA, 2002; http://www.math.ucla.edu/~ynm/papers/foundalg.pdf

Accounting for Death in Guns

Massacre by shooting is uncommon. A movie theater showing a Batman movie, Columbine High School, Fort Hood, Sandy Hook Elementary School, and Olso, Norway, together claimed 240 casualties, dead and wounded. There are undoubtedly more, and I encourage you to find every relevant mass murder committed with a gun over the past 15 years and tally the casualties. Now find the annual deaths in the U.S. attributed to, say, bicycling, choking by children 14 and younger, food poisoning, drowning, or motorcycling. The U.S. Centers for Disease Control and Prevention maintains metrics for most of them (http://remstate.com/blog/2013/04/gun-violence-facts-and-citations.html). Each year, any of these causes claims more lives than massacres involving guns. Tailoring specific solutions to these statistically insignificant occurrences is an ineffective expenditure of both resources and rights, regardless of how tragic the individual cases are. However, gun-related deaths in general are on par with total U.S. automobile-related deaths per year, ~30,000, suggesting we are able to make progress on the larger problem.

Suicide is a top cause of death in the U.S., with more than 38,000 per year, about half involving guns, accounting for two-thirds of annual gun-related deaths in the U.S. "Fixing" guns, as Jeff Johnson discussed in his Viewpoint "Can Computer Professionals and Digital Technology Engineers Help Reduce Gun Violence?" (Mar. 2013), will not fix suicide. There might be some improvement (guns are convenient and quick), but identifying and helping depressed people would be more effective.

The ~11,000 annual gun-related homicides in the U.S. (of ~16,000 homicides total) account for the final one-third of gun-related deaths. Note the CDC includes all homicides, criminal and noncriminal. The FBI provides numbers that help separate the "bad" incidents from the "good"; for example, it appears that ~5.6% of all justified homicides are noncriminal, and virtually all justifiable homicides turn out to involve guns. Justified shootings reflect a 60%/40% split between those committed by police and those by private citizens.

Good or bad, guns work as advertised—as lethal weapons.

Attempting to use technology to reduce gun lethality is destined to follow the same path of failure as digital rights management technology, with law-abiding gun owners risking failure of their weapons in potentially life-threatening situations, while criminals jailbreak theirs to sidestep "safety restrictions" intended to prevent crime.

I have left out the 606 annual accidental deaths (2010) in the U.S. involving guns. Amazingly, more people die in bicycle accidents per year (~700 in the U.S.) than by accidental shooting, speaking well for the state of gun safety today.

Consider the following three suggestions for reducing violent deaths in the U.S. (including those involving guns): help medical professionals detect and/or treat underlying causes (such as depression); apply that knowledge to keep guns away from at-risk people; and eliminate the rock star ideal (nonstop work, hitting perpetually unreasonable deadlines, monotonic achievement) from high-tech culture to reduce stress-induced depression and suicide.

Travis Snoozy, Seattle, WA

Author's Response:

Comparing frequencies of intentional crimes and accidents is nonsensical. Injuries from fights in schools happen less often than injuries from playground accidents. Should we therefore allow fights? Bombings in the U.S. are rare compared to car accidents. Should we therefore accept bombings? We put effort where potential leverage is greatest, and we have more leverage over intentional crimes than over accidents. Keep guns away from "at-risk" people through universal background checks and waiting periods. Digital rights management technology has failed? That's news to me.

Jeff Johnson, San Francisco, CA

Scenario Approach for Ethical Dilemmas

I wrote and won approval for the first ACM Professional Guidelines, which evolved into the ACM Code of Ethics in the 1970s. This led to my obtaining two National Science Foundation grants from the Office of Science and Society, Science Education Directorate, Ethics and Values in Science and Technology Program. The grants helped me hold two ACM ethics workshops in 1977 and 1987, respectively, resulting in two books, *Ethical Conflicts in Computer Science and Technology*, AFIPS Press, 1981, and *Ethical Conflicts in Information and Computer Science, Technology, and Business*, QED Information Sciences, 1990.

I strongly support Rachelle Hollander's scenario approach to ethics explored in her Viewpoint "Ethics Viewpoints Efficacies" (Mar. 2013), finding it useful and revealing in the two ACM ethics workshops. I invited CS opinion leaders to discuss, evaluate, and vote "unethical" or "not unethical" on almost 100 ethical-conflict scenarios. The scenarios came from my earlier NSF-funded studies of cases of computer abuse and misuse. It was fascinating to see how ethical values changed from 1978 to 1988. Ponder how they have changed since.

Donn B. Parker, Los Altos, CA

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

BLOG@CACM

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

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

DOI:10.1145/2461256.2461261 http://cacm.acm.org/blogs/blog-cacm

Computer Security Needs Refocus, And Be Nice About It

Jason Hong wonders whether computer security is missing the mark, while Judy Robertson supports refusing to tolerate discourtesy.

Jason Hong "Is the Computer Security Community Barking Up the Wrong Trees?"

http://cacm.acm.org/blogs/blog-cacm/144349-is-the-computer-security-community-barking-up-the-wrong-trees/fulltext
December 15, 2011

I've been saying for a while that there's a pretty big mismatch right now between what everyday people need with respect to computer security and what the computer security community, both research and industry, are actually doing.

My ammunition comes from Microsoft's Security Intelligence Report, which presents an overview of "the landscape of exploits, vulnerabilities, and malware" for the first half of 2011.

The report presents a number of fascinating findings. For example:

- Very few exploits actually use zero-day vulnerabilities. Microsoft's Malicious Software Removal Tool found no major families of vulnerabilities using zero-day attacks. Microsoft's Malware Protection Center also found that, of all exploits used, at most 0.37% of them used zero-day attacks. Here, zero-day is defined as a vulnerability where the vendor had not released a security update at the time of the attack.
- 44.8% of vulnerabilities required some kind of user action, for example clicking on a link or being tricked into installing the malware.
- 43.2% of malware detected made use of the AutoRun feature in Windows.

The reason Microsoft's report is important is because it offers actual data on the state of software vulnerabilities, which gives us some insight as to where we as a community should be devoting our resources. As one specific example, if we could teach people to avoid obviously bad websites and bad software, and if AutoRun were fixed or just turned off, we could avoid well over 80% of malware attacks seen today.

However, there's a big mismatch right now between what the data says about the vulnerabilities and what kind of research is being done and what kind of products are being offered. For example, there are at most a handful of research papers published on the user interaction side of protecting people from vulnerabilities, compared to the 500+ research papers listed in the ACM Digital Library on (admittedly sexier) zero-day attacks.

This isn't a mismatch just in computer research. Just go to any industry trade show, and try to count the number of companies that have a real focus on end users. No, not network admins or software developers, I mean actual end users. You know, the people that try to use their computers to accomplish a goal, rather than as a means toward that goal, like accountants, teachers, lawyers, police officers, secretaries, administrators, and so on. The last time I went to the RSA conference, I think my count was two (though to be honest, I may have been distracted by the sumo wrestler, the scorpions, and the giant castle run by NSA).

Now, I don't want to understate the very serious risks of popular themes in computer security research and products made by industry. Yes, we still do need protection from zero-day attacks and man-in-the-middle attacks, and we still need stronger encryption techniques and better virtual machines.

My main point here is that attackers have quickly evolved their techniques toward what are primarily human vulnerabilities, and research and industry have not adapted as quickly. For computer security to really succeed in practice, there needs to be a serious shift in thinking, to one that actively includes the people behind the keyboard as part of the overall system.

Judy Robertson "We Needn't be Malevolent Grumps"

http://cacm.acm.org/blogs/blog-cacm/144878-we-neednt-be-malevolent-grumps-in-2012/fulltext
December 31, 2011

A few months back, Bertrand Meyer wrote about the nastiness problem in computer science, questioning whether we as reviewers are "malevolent grumps." Judging by the user comments on the page, this hit a nerve with readers who were the victims of such grumpiness! Jeannette Wing then followed up on this with some numbers from NSF grant rejections that did indeed indicate that computer scientists are hypercritical. Much as I enjoy the colorful phrasing, I feel that a field full of malevolent grumps is not something we should simply accept. In fact, even if there are only a few grumps out there, it's in all our interests to civilize them.

So what can computer scientists do to reduce the nastiness problem when reviewing? Reviewers, authors, program committee members, conference chairs, and journal editors can all do their bit by *simply refusing to tolerate discourtesy*. Let's embrace the rule: We no longer ignore bad behavior. As reviewers, we can aim to be polite (yet stringent) ourselves but also to point out to co-reviewers if we find their impoliteness unacceptable. As authors, we do not have to accept a rude review and just lie down to lick our wounds. We can (politely!) raise the issue of rudeness with the program chair or editor so it is less likely to occur in the future. As editors, chairs, and program committee members, we can include the issue of courtesy in the reviewing guidelines and be firm about requesting reviewers to moderate their tone if we notice inappropriate remarks.

One of the first steps is to separate intellectual rigor from discourtesy. It is possible to be critical without being rude or dismissive. We can maintain standards in the field without resorting to ill-natured comments. (Believe it or not, it is also possible to ask genuine questions at a conference without seeking to show off one's own intellectual chops, but that is another matter). The purpose of reviewing, in my view, is to help an author improve their work, not to crush them under the weight of your own cleverness. It's not the author's fault that you had a bad day, or that some other reviewer just rejected your own paper.

Of course, there are some pockets of good reviewing practice within the field that we can draw on. I am sure there are many, but I have chosen CHI because I have been writing for it recently. The CHI conference is one of the biggest, most well-respected annual human computer interaction conferences. In 2011, there were 2,000 attendees from 38 countries. This year there were 1,577 paper submissions with a 23% acceptance rate. This was the first year I submitted papers to it, and I have been impressed by the quality of the reviews in terms of their fairness, constructiveness, and level of detail. They contained greater insight and intellectual oomph than the reviews I had from a high-impact journal recently. For one of my CHI submissions, the reviewers did not agree with the paper on some points—it is on a controversial topic—but they still offered suggestions for how to resolve these issues rather than simply rejecting the paper. Was I just lucky in the reviewers I was allocated? Possibly, but the CHI reviewing process has some interesting features built in to maintain review quality.*

1. In the guidelines for reviewers, courtesy is explicitly mentioned: "**please be polite** to authors. Even if you rate a paper poorly, you can critique it in a positive voice. As part of polite reviewing practice, you should always **state what is good about a paper first**, followed by your **criticisms**. If possible, you should offer **suggestions for improvement** along with your criticism."

2. Authors can select both the subcommittee and the contribution type for a paper, which maximizes the chance that the paper will end up with reviewers with appropriate expertise, and that the reviewers will use criteria appropriate to the paper when assessing its suitability (e.g., not insisting on empirical evidence for a theoretical contribution).

3. The reviewing process is thorough and has several opportunities for unfairness or discourtesy to be weeded out. Each paper is blind-reviewed by three or more experts, and then an associate chair writes a meta-review to summarize the assessment of the paper, and what action (if any) should be taken to improve it. In this way, individual grumpiness is moderated. A variant of this good practice from other conferences is when reviewers of the same paper can see each other's reviews (once they have submitted their own), thus introducing peer pressure not to be awful.

4. Authors have a right to reply by writing a rebuttal of the review. The rebuttal is taken into account along with a revised meta-review (and potentially revised individual reviews) at a two-day committee meeting when final accept/reject decisions are made.

5. All submitting authors are surveyed about their opinions of the reviewing process—yet another chance to raise issues about unfairness or discourtesy that have not been addressed in a rebuttal.

6. This point is more about the nature of the conference itself, rather than the reviewing procedures. Because CHI is so interdisciplinary, participants have a wide range of backgrounds from art and design to hardcore engineering. They are therefore exposed to—and may in fact seek out—different perspectives that may make them open to different paradigms as reviewers. Could colleagues from the arts and social sciences be having a civilizing influence on the grumpy computer scientists?

This is a fairly heavyweight process, but if conference organizers adopted even just one more of the practices from points 1–5, or if journal editors added a courtesy clause to their review instructions, the world would be a slightly better place.

* Thanks to Tom Erickson–the person who runs the CHI author survey–for kindly raising some of these points.

Jason Hong is an associate professor of computer science at Carnegie Mellon University. **Judy Robertson** is a lecturer at Heriot-Watt University.

Come join us for the 19th annual Reflections | Projections conference in Urbana-Champaign, Illinois! From October 10th - 13th, you can hear talks from great speakers across industry and academia, participate in exciting competitions, and attend our career fairs! Hosted by the ACM student chapter at the University of Illinois, Reflections | Projections is one of the largest student-run conferences in the US. The event is free and open to everyone, regardless of university affiliation, but please pre-register at the link below for a free t-shirt!

photo credit Priten Vora

OCT 10

Startup Fair:
Meet all of the small companies who are hard at work solving big problems!

OCT 11

Job Fair:
Interested in the industry leaders like Microsoft or Facebook? Check out this larger career fair. Dress casual!

Speaking Events:
Friday night includes two speaking events and dinner, so come have some fun!

Mechmania Opening Ceremony:
Our 24-hour programming competition starts Friday night. Work on getting your client ready to compete in the tournament.

OCT 12

More Speaking Events:
The bulk of our talks occur all throughout Saturday. Come see just one or all of them.

Puzzlebang Finishes:
Our week-long puzzle competition wraps up! Let's find out who won.

OCT 13

Mechmania Closing Ceremony:
See the final round of contestants and the awards ceremony!

reflections | projections **2013**

acm.uiuc.edu/rp

Science | DOI:10.1145/2461256.2461262

Gary Anthes

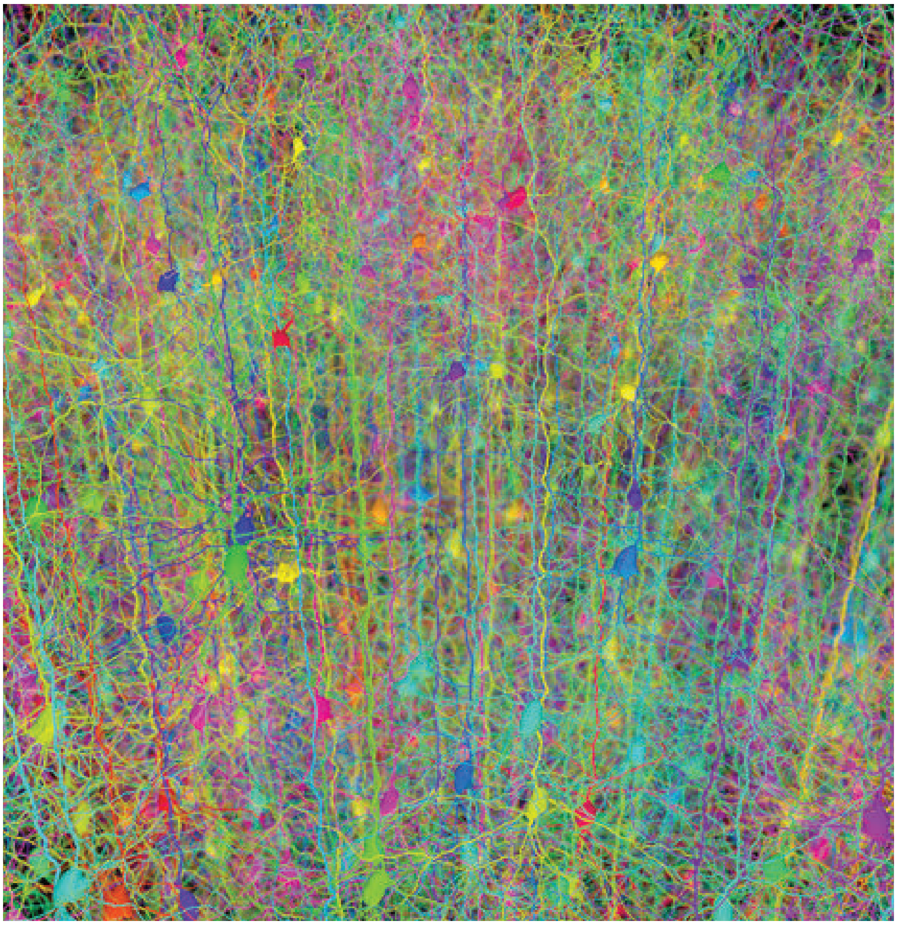
Deep Learning Comes of Age

Advances on multiple fronts are bringing big improvements to the way computers learn, increasing the accuracy of speech and vision systems.

IMPROVEMENTS IN ALGORITHMS and application architectures, coupled with the recent availability of very fast computers and huge datasets, are enabling major increases in the power of machine learning systems. In particular, multilayer artificial neural networks are producing startling improvements in the accuracy of computer vision, speech recognition, and other applications in a field that has become known as "deep learning."

Artificial neural networks ("neural nets") are patterned after the arrangement of neurons in the brain, and the connections, or synapses, between the neurons. Work on neural nets dates to the 1960s; although conceptually compelling, they proved difficult to apply effectively, and they did not begin to find broad commercial use until the early 1990s.

Neural nets are systems of highly interconnected, simple processing elements. The behavior of the net changes according to the "weights" assigned to each connection, with the output of any node determined by the weighted sum of its inputs. The nets do not work according to hand-coded rules, as with traditional computer programs;



Rainbow brainwaves made from a computer simulation of pyramidal neurons found in the cerebral cortex.

IMAGE BY HERMANN CUNTZ

they must be trained, which involves an automated process of successively changing the inter-nodal weights in order to minimize the difference between the desired output and the actual output. Generally, the more input data used for this training, the better the results.

For years, most neural nets contained a single layer of "feature detectors" and were trained mainly with labeled data in a process called "supervised" training. In these kinds of networks, the system is shown an input and told what output it should produce, such as letters of the alphabet. (In unsupervised learning, the system attempts to model patterns in the unlabeled input without knowing in advance what the desired outputs are.) However, labeling data, particularly when there are many possible values, is labor-intensive and slow. Explains machine learning pioneer Geoffrey Hinton, a computer scientist at the University of Toronto, "The basic approach back then was, you hand-engineered a bunch of features, and then you learned what weights to put on the features to make a decision. For example: if it's red, it's more likely to be a car than a refrigerator."

In the 1980s, Hinton and others came up with a more powerful type of supervised learning, one that employed learning in multiple layers, combining low-level features into successively higher levels. However, Hinton says that with a few exceptions, these systems did not work as well as expected. The process of starting with very low-level features, such as the intensities of individual pixels, and learning multiple layers of features all at the same time, involved a huge amount of computation; computers were not fast enough, there was not enough labeled data, and system developers did not have a good way to initialize the weights.

Recent Developments

Since about 2005, the picture has changed dramatically. Hinton (with Yann LeCun, a professor of computer and neural science at New York University, and others) made a number of fundamental advancements in neural nets, principally with unsupervised learning and multilayer learning. Neither concept was new, but improved algorithms, more data, and faster computers make them work much better than early versions. Of unsupervised learning, Hinton says, "That's much more like what people do—you are not given labels. It's much easier to get unlabeled training data; just take a video camera and wave it at the world."

"A wave of excitement today comes from the application of unsupervised learning to deep neural nets."

While the idea of unsupervised learning had obvious appeal for years, it had not been clear how to put it inside a hierarchical, multilayer system, says research collaborator LeCun. "You want four or five or six layers, because that's how you go from edges to textures to parts of objects to whole objects in particular configurations to whole objects regardless of configuration." And that is conceptually how the brain works, research has shown.

"A wave of excitement today comes from the application of unsupervised learning to deep neural nets," LeCun says, adding that another wave of excitement surrounds the use of the more-traditional supervised training of multilayer systems called "convolutional" neural nets. LeCun developed convolutional neural nets at Bell Laboratories in the late 1980s; they were among the earliest to employ multilayer learning.

Convolutional nets, a biologically inspired model for image recognition, can recognize visual patterns directly from pixel images with minimal preprocessing. The system processes a small—say 10 by 10 pixels—portion of an image, looking for small features such as edges, then slides the 10-by-10 window over one pixel and repeats the operation until the entire image has been processed. It produces output values that are sums of the inputs weighted by the synaptic weights in each small window.

The reason these nets succeeded in an era of relatively weak computers is that by working on a small window—100 pixels, say—instead of millions of pixels, the number of connections and weights, and hence the computational workload, was greatly reduced.

Today, faster computers, more data and some "simple architectural tricks" applied by LeCun have allowed multilayer convolutional neural nets to become practical with unsupervised learning, where the net is trained, one layer after another, using only unlabeled data. The process involves initial unsupervised training to initialize the weights in each layer of the network, followed by a supervised global refinement of the network. "This works very well when you have very fast machines and very large datasets, and we have just had access to those recently," LeCun says. Convolutional neural nets are well suited to hardware implementations, he says, and we will see many embedded vision systems based on them in the coming years.

This approach — initializing weights in each layer at first using a large amount of unlabeled data, followed by fine-tuning the global net using a smaller amount of labeled data — is called "semi-supervised" learning. "Before, if you initialized purely randomly, the systems would get lost," says John Platt, manager of the Machine Learning Groups at Microsoft Research (MSR). "This gives the network a shove in the right direction."

Results

The use of semi-supervised learning and deep neural nets is the basis for some of the more dramatic results seen recently in pattern recognition. For 20 years, most speech systems have been based on a learning method that does not use neural nets. In 2011, however, computer scientists at MSR, building on earlier work with the University of Toronto, used a combination of labeled and unlabeled data in a deep neural net to lower the error rate of a speech recognition system on a standard industry benchmark from 24% to about 16%. "Core speech recognition has been stuck at about 24% for more than a decade," Platt says. "Clever new ideas typically get a 2% to 5% relative improvement, so a 30% improvement is astounding. That really made the speech people sit up and take notice."

In last year's ImageNet Large Scale Visual Recognition Challenge, Hinton's team from the University of Toronto scored first with a supervised, seven-layer convolutional neural network trained on raw pixel values, utilizing two NVIDIA graphics processing units (GPUs) for a week. The neural network also used a new method called "dropout" to reduce overfitting, in which the model finds properties that fit the training data but are not representative of the real world. Using these methods, the University of Toronto team came in with a 16% error rate in classifying 1.2 million images, against a 26% error rate by its closest competitors. "It is a staggeringly impressive improvement," says Andrew Zisserman, a computer vision expert at the University of Oxford in the U.K. "It will have a big impact in the vision community."

Also last year, researchers at Google and Stanford University claimed a 70% improvement over previous best results in a mammoth nine-layer neural network that learned to recognize faces without recourse to any labeled data at all. The system, with one billion connections, was trained over three days on 10 million images using a cluster of machines with a total of 16,000 cores.

The different models for learning via neural nets, and their variations and refinements, are myriad. Moreover, researchers do not always clearly understand why certain techniques work better than others. Still, the models share at least one thing: the more data available for training, the better the methods work.

MSR's Platt likens the machine learning problem to one of search, in which the network is looking for representations in the data. "Now it's much easier, because we have much more computation and much more data," he says. "The data constrains the search so you can throw away representations that are not useful."

Image and speech researchers are using GPUs, which can operate at teraflop levels, in many of their systems. Whether GPUs or traditional supercomputers or something else will come to dominate in the largest machine learning systems is a matter of debate. In any case, it is training these systems with large amounts of data, not using them, that is the computationally intensive task, and it is not one that lends itself readily to parallel, distributed processing, Platt says. As data availability continues to increase, so will the demand for compute power; "We don't know how much data we'll need to reach human performance," he says.

Hinton predicts, "These big, deep neural nets, trained on graphics processor boards or supercomputers, will take over machine learning. The hand-engineered systems will never catch up again."

Further Reading

Le, Q. and seven others
Building High-level Features Using Large Scale Unsupervised Learning, ***Proceedings of the 29th International Conference on Machine Learning***, **Edinburgh, Scotland, 2012**

Dahl, G., Yu, D., Deng, L., and Acero, A.
Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition, draft accepted by ***IEEE Transactions on Audio, Speech, and Language Processing***, **http://research.microsoft.com/pubs/144412/dbn4lvcsr-transaslp.pdf**

Hinton, G.
Brains, Sex, and Machine Learning, GoogleTechTalks (video), June 22, 2012 http://www.youtube.com/watch?feature=player_embedded&v=DleXA5ADG78#!

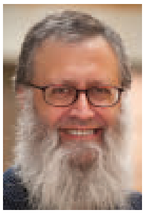
Krizhevsky, A., Sutskever, I., and Hinton, G.
ImageNet Classification with Deep Convolutional Neural Networks, paper to appear in ***Proceedings of the Neural Information Processing Systems Foundation 2012 conference***, **Lake Tahoe, NV http://www.image-net.org/challenges/LSVRC/2012/supervision.pdf**

LeCun, Y., Kavukvuoglu, K., and Farabet, C.
Convolutional Networks and Applications in Vision, ***Proc. International Symposium on Circuits and Systems***, **IEEE, 2010 http://yann.lecun.com/exdb/publis/pdf/lecun-iscas-10.pdf**

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

In Memoriam

David Notkin, 1955–2013



David Notkin, Professor and Bradley Chair of the Department of Computer Science & Engineering (CSE) of the University of Washington (UW), and associate dean of Research and Graduate Studies in UW's College of Engineering, died April 22 at the age of 58.

Notkin's research focus was in software engineering; as he explained on the UW website, "understanding why software is so hard and expensive to change, and in turn reducing those difficulties and costs." These interests underscored Notkin's belief "that the ability to change software—that is, the 'softness' of software—is where its true power resides."

Eugene Spafford, chair of the ACM Public Policy Council, said Notkin "was the personification of 'considerate.' He was deeply thoughtful about everything he did and the people with whom he came in contact. He cared if people could succeed. He cared that tomorrow will be better than today, for more than himself. And most of all, he acted on his caring in ways that did, indeed, make a difference for many, many people."

Colleague and friend Ed Lazowska, Bill & Melinda Gates Chair in Computer Science & Engineering at UW, said Notkin was an accomplished researcher, but "he was all about people and relationships—his family, his friends, his professional colleagues, and most importantly, his students. At the University of Washington, and more broadly in the field, he was our compass: he could always be counted upon to point us in the right direction."

For more information on Notkin, visit the web page of Notkinfest (http://news.cs.washington.edu/2013/02/01/honoring-david-notkin/), an event held in February 2013 at UW CSE honoring Notkin for his contributions to the field of computer science, and to announce a fellowship in his name.

Technology | DOI:10.1145/2461256.2461263

Gregory Mone

The Future Is Flexible Displays

Manufacturers hint that bendable screens are coming soon, but academics argue that many engineering challenges remain.

This past January at the Consumer Electronics Show (CES) in Las Vegas, Samsung senior vice president Brian Berkeley offered a look at the future of smartphones and tablets. During a keynote address, Berkeley showed off a prototype smartphone sprouting a thin, bright, and completely flexible display that slid out like a credit card from a wallet. The crowd clapped and cheered while he bent the display as if it were a playing card. A moment later, he reached into his jacket pocket and revealed a device that looked more like a market-ready product than a *Star Trek* gadget. The prototype resembled a standard smartphone, but the screen wrapped around the sides of the device. This way, Berkeley explained, important messages and updates could be displayed along the edges, like a ticker.

A prototype of Samsung's flexible smartphone, shown at the International Consumer Electronics Show in January 2013.

PHOTOGRAPH BY JAE C. HONG/AP IMAGES

Samsung also offered a video preview of a few even-more-science-fiction-style devices, including a smartphone that opens like a book, transforming into a seamless tablet on the inside, and a cylindrical gadget with a roll-out screen. The presentation was stunning, but also short on details. Berkeley noted that these devices would be enabled by the YOUM brand of flexible organic light-emitting diode (OLED) displays. He speculated that the technology would allow designers to come up with an entirely new ecosystem of devices. However, Samsung did not reveal any potential release dates or many technical specifications.

In fact, the prototypes were not actually functioning phones, and skeptics were quick to note that this was hardly the first time a company had heralded the imminent arrival of bendable displays. Philips revealed a rollable prototype screen in 2005. Nokia introduced its own innovative variation in January 2011. LG, Sharp and other manufacturers have shown off bendable screen technology themselves. Still, there is a growing sense that the latest prototypes are not mere trade show smoke and mirrors, and that the futuristic displays will be moving into the marketplace within the next year, and possibly sooner.

Whether these first actual product iterations prove as exciting as the fully bendable, high-resolution prototypes is another question. Academic experts are skeptical, noting that numerous technological and economic hurdles remain. "You can go to a company now and see this stuff and use it and touch it. We are tantalizingly close," says materials scientist Andrea Ferrari of the University of Cambridge. "But there's a huge gap between the lab and the devices on the market."

Engineering Challenges

The basic technology behind the flat displays in today's phones and tablets is a sandwich of different material layers. An OLED screen, for example, typically starts with a glass substrate, on top of which is a circuit containing thin-film transistors and a capacitor, then the light-emitting OLED layers and, finally, a transparent, protective layer on top. "A flat panel display screen is a very complicated product," says Nick Colaneri, director of Arizona State University's Flexible Display Center. Trying to transform these rigid surfaces into bendable devices only deepens the complexity. "The support system in every flat panel display is a piece of glass, and to make a flexible display, the first thing you have to do is get rid of the glass."

Colaneri says this essentially requires revising the way companies have built these electronics for years. Samsung and others have already begun to tackle the problem. The Samsung prototypes rely on a new kind of OLED technology in which the rigid glass substrates are replaced by flexible plastic. Yet relying on OLEDs also creates additional challenges, Colaneri says. OLEDs are hypersensitive to oxygen and water, which is not an ideal feature for a consumer product. "They need to be hermetically sealed," he explains, "and figuring out a low-cost hermetic seal is a huge problem."

Of course, OLED displays are not the only option. The basic electrophoretic technology behind E-Ink-powered devices, for example, has always been flexible. The catch is that this approach will not achieve the speed or color fidelity of an OLED display, and the electrodes that turn each of the millions of microcapsules either black or white have typically been made using a rigid substrate. Now, however, E Ink has developed a plastic-based thin film transistor that can be laminated to its microcapsule display technology. "The display is ready for implementation," says Jenn Vail, senior marketing manager at E Ink. "We're now working with partners interested in launching new devices."

Relying on plastic substrates might require some unfortunate trade-offs. Due to ever-increasing performance needs, display manufacturers may soon be shifting from amorphous silicon, the material of choice for transistors, toward amorphous oxide semiconductors (AOS) or indium gallium zinc oxide (IGZO) semiconductors. At CES in Las Vegas, Sharp Electronics unveiled a line of stunningly clear IGZO-based screens that could also be curved and bent.

The switch is due, in part, to the fact that thin-film transistors that use AOS will be better equipped to meet the performance needs of tomorrow's devices. The problem, says John Wager, an electrical engineer at Oregon State University, is that much like their predecessors, AOS transistors run best at higher temperatures. To be compatible with a flexible plastic substrate, which is more susceptible to melting than glass, the process temperature needs to drop, and that translates into lesser performance. "You can make transistors, but they're pretty poor," he says.

As a result, Wager is holding out hope that companies like Corning will soon develop an affordable flexible glass that might allow for higher temperatures. In 2012, Corning announced its thin, slightly bendable Willow Glass line, but the company recently explained that truly flexible electronics applications are still several years away.

The Limbo flexible concept smartphone, made to be used as a standard handset or bent around the arm and worn in wristwatch fashion.

SMARTPHONE CONCEPT BY JEABYUN YEON

Potential Applications

Due to all the unresolved questions and technological hurdles, Colaneri expects the first generation of flexible displays to be closer to the immobile wraparound screen Berkeley presented in January than the one he bent like a playing card. In fact, when manufacturers talk about "flexible" screens, they are talking about devices with a range of possible features. At one end of the spectrum lies the version that bends just enough so that it is resistant to shattering, and at the opposite end you have the high-resolution, laptop-sized screen that rolls up tight.

At Arizona State's Flexible Display Center, for example, Colaneri notes that the goal was never really to push roll-up screens. The mission, he says, was to further the development of lightweight, low-power devices that do not quit functioning when you drop them. Typically, a dropped device does not break because the glass screen cracks; it stops working because of damage to the glass-based transistor layer buried within. Devices with these features are already becoming available. The Wexler Flex ONE, an e-reader based on LG technology, relies on a plastic substrate so it bends just enough to survive being crammed in your pocket, and resists damage from accidental drops.

E Ink is touting its own flexible technology for the same reason. Although the company says it could be used to develop roll-up-style e-newspapers, the real breakthrough may be in creating accident-proof gadgets. This could prove critical for several of the applications E Ink has in its sights, including electronic textbooks for students, a more drop-prone group than adults. "We were also able to make material improvements to increase the ruggedness of the display connectors," Vail adds.

Flexible screens might also assume forms engineers and designers have not yet envisioned. For example, a pair of ex-IDEO designers recently raised money through a Kickstarter campaign for an E-Ink-based flexible-screen watch called the CST-01.

Colaneri hints that it could take a few years before flexible displays truly find their appropriate niche. "The black-and-white screens that became e-readers started out in lots of other applications before people figured out they wanted e-readers," says Colaneri. "Flexible screens are in that ugly adolescence right now."

Whether they mature quickly enough to deliver the roll-out of futuristic, highly bendable screens in the next few years is an open question. Colaneri suspects that vision is still several years away. "Getting to a phone that bends as much as a credit card would before I break it? That's a ways out there yet."

Further Reading

Park, Jin-Seong, Chae, Heeyeop, Chung, Ho Kyoon, Lee, Sang In.
Thin Film Encapsulation for Flexible AM-OLED: A Review. *Semiconductor Science and Technology*, February 2011.

Kim, Sunkook, et. al.
Flexible Displays: Low Power Organic Light-Emitting Diode Display Device. *Advanced Materials*, August 2011.

Crawford, Gregory, editor.
***Flexible Flat Panel Displays*. Wiley 2005.**

Sharp Electronics USA.
IGZO: a video intro to indium-gallium zinc oxide. http://youtu.be/SnUUXoFsjoY

Samsung.
Samsung Flexible Display at CES 2013: 'YOUM.' http://youtu.be/N3E7fUynrZU

Gregory Mone is a Boston, MA-based writer and the author of the novel Dangerous Waters.

Society | DOI:10.1145/2461256.2461264

Paul Hyman

Augmented-Reality Glasses Bring Cloud Security Into Sharp Focus

The possibility of a new $200-billion-plus industry has cloud security experts bracing for the ramifications.

IF CLOUD SECURITY is an issue to be reckoned with today, the problem will only worsen as more and more data is saved and backed up to the cloud, say experts.

Indeed, a new consumer product being developed by such players as Google, Microsoft, Apple, and others will likely generate more data—perhaps by an order of magnitude—than today's smartphones and media tablets combined.

Augmented-reality (AR) glasses, also known as "wearable computers," are designed to display information hands-free in smartphone-like format and to interact with the Internet via natural language voice commands.

According to Matthew Green, assistant research professor at the Johns Hopkins Information Security Institute, AR glasses "will be collecting everything you see, everything you say—and potentially not just backing up all that information but sending it in real time for services like Google to process and to respond with relevant information. This is a big leap in data collection."

Of course everything depends on how popular the devices become, but Google's version—branded as Google Glass—already has been dubbed by *Time* magazine one of the best inventions of 2012, even though it is not expected to be available to the public until 2014. Meanwhile, last July, Apple applied for a patent for its own version of AR glasses that the press is calling "iGlasses," and, four months later, Microsoft did the same.

All three prospective competitors were contacted to comment on their projects and their business models; all three declined to be interviewed.

Google co-founder Sergey Brin, wearing Google Glass.

PHOTOGRAPH BY CHRIS CHABON

Microsoft's device, of which there is no public prototype, seems to be a bit less ambitious than Google Glass, according to a recent TechCrunch article. Rather than being intended for all-day use, the Microsoft glasses are designed for use in a stationary position, such as at a baseball game, where the glasses might display scores, pitch speeds, and other information overlaid on the wearer's view.

> **AR glasses "will be collecting everything you see, everything you say."**

By all indications the Microsoft glasses are not yet in production, but there is speculation that should they come to market, they might also plug into existing Microsoft hardware, perhaps to produce combined gaming experiences with Xbox and Kinect.

Similarly, Apple's device is also under wraps. Its patent indicates it is designed to incorporate a head-up display (HUD) in front of both eyes—not just one, like Google Glass—and to connect to an external device with a 16:9 aspect ratio, possibly an iPhone or TV.

Johns Hopkins' Green believes the main reason the competition to develop an AR glasses technology has become so hot and heavy is the same reason other mobile technologies—like smartphones and media tablets—have become so profitable.

"The commercial value of the glasses is to enable these companies to sell more services and products to consumers," he says. "The glasses will collect data about where you are and what you're seeing, then filter that through search engines like Google or Bing, and ultimately respond with useful information—along with targeted advertising. For a company like Google, for example, which is one of the world's largest advertising companies, it sure makes a lot of sense for them to be in that space."

Green, a computer security expert, is mainly concerned that the companies will use a portion of the data these products capture in whatever way is useful to them today, but also that they will retain that data for analysis later on, for use in ways that perhaps they have not thought of yet.

"That introduces another risk," Green says, "which is that all your data is sitting on servers waiting for somebody to steal it. After all, the cloud is just a lot of computers in data centers and, while they may use best-of-breed technology, any computer security expert will tell you that, unfortunately, nothing is completely secure."

At BT, chief security technology officer Bruce Schneier believes the risks of cloud storage—a topic he frequently blogs about—are already considerable, and the popularity of AR glasses will increase those risks only slightly. To Schneier, AR glasses are no different from any other product or application that stores data in the cloud.

"Almost everybody has all their e-mail going through the cloud. Many people store their files in the cloud. Your address book is there, your calendar is there, all your socialization like Facebook is there, as is your location and your phone information," he says. "The real worries are not about any one thing, but about the totality of everything."

As a security expert, Schneier admits it is impossible to know the extent to which the data in the cloud may be at risk, because the security issues are social, not technological, having to do more with laws and social norms.

"Facebook has your data because you gave it to them," he says, "and the law says they can do whatever they want with it. So, it doesn't matter what your technology is or what you use to stop misuse of that data. The answer is not to give them the data in the first place."

Yet AR glasses are likely to be the "next big thing" and have the makings of becoming a $240-billion-plus industry, according to Steve Mann, a professor at the University of Toronto's Department of Electrical and Computer Engineering and the general chair (and a keynote speaker) at this month's IEEE 2013 International Symposium on Technology and Society. The conference's theme is "the social implications of wearable computing and augmented reality in everyday life."

"Life-glogger" Steve Mann and his EyeTap Digital Eye Glass. Mann has been wearing some version of this since 1978.

PHOTOGRAPH COURTESY OF STEVE MANN

No one is likely more familiar with the technology than Mann, having invented AR glasses—which he calls his EyeTap Digital Eye Glass—back in 1978 to assist the visually impaired, and then attached a set to his head permanently, which he has worn ever since.

In the 1980s, Mann came up with what he called "life-glogging," capturing and streaming his life 24/7 to the Internet by bringing his own infrastructures with him wherever he went. When he traveled to different countries, Mann updated his radio license to operate in that location and put his servers on the rooftops of tall buildings to permit wireless connectivity. He migrated the project to the World Wide Web in the early 1990s, and started a community of life-gloggers that has grown to more than 200,000 users.

"It's become a very interesting research project that's generated a lot of interesting conversation," he observes.

Milestones

IJCAI-13 Names 3 For AI Awards

The International Joint Conference on Artificial Intelligence (IJCAI-13) recently named the recipients of several prestigious awards.

The IJCAI 2013 Award for Research Excellence, given annually to a scientist who has carried out a program of research of consistently high quality, yielding substantial results, was awarded to Hector Levesque, professor of computer science at the Department of Computer Science of the University of Toronto, in recognition of his work on knowledge representation and reasoning, including cognitive robotics, theories of belief, and tractable reasoning.

IJCAI's 2013 Computers and Thought Award, presented to outstanding young scientists in artificial intelligence, has been bestowed upon Kristen Grauman, associate professor in the Department of Computer Science, University of Texas at Austin, in recognition of her fundamental contributions to recognition and search problems in computer vision.

The Donald E. Walker Distinguished Service Award, honoring senior scientists in AI for contributions and service to the field, will be presented at IJCAI-13 in Beijing in August, to Wolfgang Wahlster, CEO and scientific director of the German Research Center for Artificial Intelligence and professor at the Saarland University, in recognition of his substantial contributions and extensive service to the field of artificial intelligence throughout his career.

Meanwhile, subsequent technological advances may mimic the public's fondness for replacing their intrusive and unfashionable glasses with contact lenses. Indeed, computer researchers at Ghent University in Belgium have built an LCD screen into a contact lens using conductive polymers and molding them into a very thin, spherically curved substrate with active layers.

At the moment, all the lens does is flash a dollar sign, but Ghent Ph.D. student Jelle De Smet and his team foresee the lens could function as an HUD that could superimpose an image onto the user's normal view. This kind of screen-on-the-eye technology could displace smartphones as the dominant way people access the Internet and connect to each other.

De Smet describes the lenses as providing information in ways similar to how Google Glass operates, but without having to wear glasses, which some people do not like to do.

"The functionality we foresee could comprise reading email and text messages, turn-by-turn directions, information about your surroundings, or a work situation where your hands need to be freed up, such as patient information for surgeons or a diagram for astronauts doing repairs on a satellite," De Smet says.

He anticipates it will take another 10 years before there will be a prototype with an acceptable number of pixels.

Regarding cloud security, De Smet says, "Each wireless technology is potentially prone to security issues," he says. "It just depends on how well your encryption techniques work."

Johns Hopkins security expert Matthew Green sees two potential problems ahead. The first is that, even with the best technology, computer scientists have been unable to stop hackers from periodically stealing data. The more relevant issue, he says, is that the companies selling the glasses are specifically designing them to provide themselves with full access to all data.

"In other words, there's no way to hide the data because these companies are the ones that are processing it," he explains, "and they are doing it for free in exchange for getting your information. As long as you allow their systems to have access to it, there is no computer science answer to that problem."

> **"Each wireless technology is essentially prone to security issues. It just depends on how well your encryption techniques work."**

Steve Mann calls that a valid concern for consumers, who ought to be worried about the integrity of their own data.

According to Wikipedia, Mann last year "teamed up with the IEEE and the ACLU" to generate support for a Mann-Wassel Law that would be presented to the New York State Legislature. However, that proposal, which focuses on security and privacy issues surrounding individuals' use of recording technologies (including AR glasses) for "sousveillance" (the recording of an activity by a participant in the activity) has not made any progress.

"We certainly have opinions about how companies ought to treat their customers' data and their privacy," says Jay Stanley, a senior policy analyst at the ACLU, "and we might be willing to support legislation as time goes on as situations warrant it. But we don't agree the legislation is ready to be proposed. There are some very complicated issues here, and we need to do more thinking on them before we would be in a position to propose legislation."

An IEEE spokesperson says the professional association "will not comment at this time" and that "IEEE has not taken an official position on this pending legislation."

In the meantime, Mann suggests making the security issues known to consumers, who can then choose to purchase from the company that makes strong security a selling feature.

"Let the market determine which brand is most successful, perhaps by being the product that is most secure," he says.

He has this recommendation for entrepreneurial computer scientists: begin thinking about secure servers and services that can be offered if AR glasses become popular.

"It might be necessary to have a secure program running on the glasses that encrypts the data before uploading it to the cloud," he suggests. "So you either have to buy glasses that do that—and I'm sure some manufacturers will take that more seriously than others—or there could be third-party providers that might offer services associated with the glasses. Eventually astute customers may seek out and choose an operating system, like Unix, and an encryption protocol that is secure."

Further Reading

"Cloud Computing," a blog by Bruce Schneier, published June 4, 2009 at http://www.schneier.com/blog/archives/2009/06/cloud_computing.html

"Feudal Security," a blog by Bruce Schneier, published December 3, 2012 at http://www.schneier.com/blog/archives/2012/12/feudal_sec.html

"Design and Wrinkling Behavior of a Contact Lens With an Integrated Liquid Crystal Light Modulator," a paper by J. De Smet, A. Avci, Roel Beernaert, Dieter Cuypers, and Herbert De Smet, published in May, 2012 in the *Journal of Display Technology* at http://8.18.37.105/jdt/abstract.cfm?uri=jdt-8-5-299 (abstract)

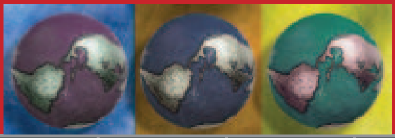
"A liquid crystal-based contact lens display," a video posted Oct. 31, 2012 by the Centre for Microsystems Technology, Ghent University, Belgium at https://www.youtube.com/watch?v=-btRUzoKYEA

"Through The Glass, Light," an article by Steve Mann, published Fall 2012 in *IEEE Technology and Society* at http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6313625

"Steve Mann: AR eyeglass and wearable computing," a video posted December, 2012 by Steve Mann at http://vimeo.com/56092841

"Wearable Computing: A First Step Toward Personal Imaging," an article by Steve Mann, published in *Computer*, Vol. 30, No. 2, February 1997 at http://eyetap.org/wearcomp/ieeecomputer/r2025.htm

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



Milestones | DOI:10.1145/2461256.2461265

Neil Savage

Proofs Probable

Shafi Goldwasser and Silvio Micali laid the foundations for modern cryptography, with contributions including interactive and zero-knowledge proofs.

Throughout history, people have used ciphers and cryptography to hide information from prying eyes. Yet it was not until 1983, when graduate students Shafi Goldwasser and Silvio Micali, the recipients of the 2012 ACM A.M. Turing Award, published their paper "Probabilistic Encryption," that cryptographers actually defined what they were doing.

"They were the first to ask the question, 'What exactly is all this trying to accomplish? What is the goal more precisely than, 'I want to hide my data'?'" says Mihir Bellare, professor of computer science and engineering at the University of California, San Diego, and a former Ph.D. student under Micali at the Massachusetts Institute of Technology (MIT); he is also a coauthor of several research papers with both. "It's in some ways amazing to think that in 2,000 years, people have looked at this and they had never seriously asked themselves what it means to break a cryptographic system."

In developing a formal definition of security, the two came up with the idea that an encryption method would be secure if an adversary given two encrypted messages could not have the slightest probabilistic advantage in distinguishing them from each other, even knowing what the two messages contained. They labeled this concept "semantic security," saying it incorporates the idea that even partial information about messages, even relationships between messages, should be hidden by their encryption.

"We designed randomized encryption methods for which you can prove that the time it would take an adversary to learn any information about encrypted messages is no less than the time it would take him to solve a mathematical problem believed to be intractable, such as factoring large numbers," says Goldwasser, a professor in the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT and at the Weizmann Institute of Science in Israel.

"A lot of the work prior to theirs just was not very rigorous," says Ronald Rivest, a fellow professor at CSAIL and a 2002 Turing Award co-recipient for his work developing the RSA method of public key cryptography. "It was heuristic in character. It didn't have a very precise definition of what security meant." Security is tricky, he says, because it is often a negative property; it's not about showing that you can do something, but about showing that someone else cannot do it.

Together with Charles Rackoff of the University of Toronto, Goldwasser and Micali came up with interactive proofs, a way to demonstrate that a person has proven a theorem to another person without the other person learning how to prove the theorem. "Using interactive proofs, you can actually get convinced with overwhelming probability of something that would have taken too long to prove with classical proofs," says Micali, also a CSAIL professor.

An interactive proof may be thought of as a game between a prover and a verifier that involves a small number of moves. If the theorem is correct, the prover wins every game; if it's false, the prover wins less than half the time. "Clever interactive proofs can convince me that things are correct without involving me in too-time-consuming work," Micali says.

"A lot of the work prior to theirs just was not very rigorous."
—Ron Rivest

Interactive proofs have led to wider study of new ways to write and check proofs. One such method, probabilistically checkable proofs, requires checking only some fraction of the proof chosen at random; if the checked portion is correct, that strongly suggests the whole proof is correct. If it is incorrect, there is a good probability that other spots checked would be wrong, too.

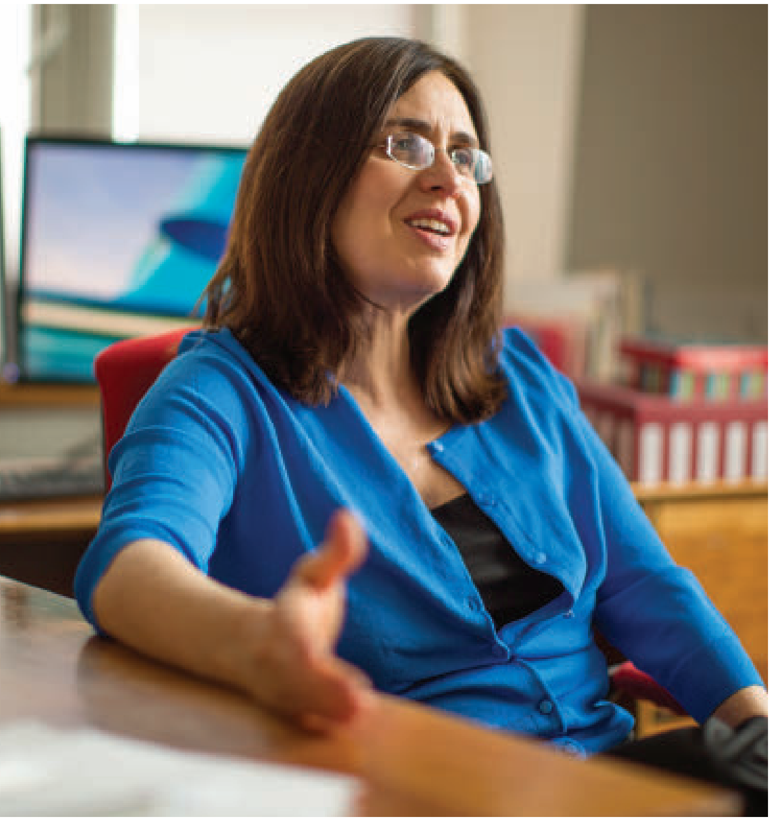
Goldwasser, Micali, and Rackoff also developed the concept of zero-knowledge proofs, which allow the user to convince someone else of the correctness properties of a piece of information, without actually revealing what that information is. A person using zero-knowledge proofs could demonstrate that she was authorized to access a secure system, without revealing her secret identifying information; zero-knowledge proofs could be used to verify that a tally of encrypted votes was computed correctly, without showing how each individual voted. "You can convince me a statement is correct without telling me why," Micali says. That ability would make it easier for multiple parties, such as businesses with sensitive financial information, to work together without revealing their secrets. "It's a way to enhance the ability of people to interact," he says.

"I can't tell you how surprised I was when I first heard the concept of zero-knowledge," says Rivest.

That was not their only unexpected contribution to the field, he says. "The necessity of using randomness in encryption was surprising to many people."

Still, it's important. To start with, the person encrypting the data picks his approach at random, thus denying the adversary the ability to know when the same message is being encrypted and sent again. In interactive proofs,

PHOTOGRAPH BY BRYCE VICKMARK



the two parties trade questions and answers, with the verifier trying to catch the prover in a lie. Because the verifier tosses a coin to choose his questions at random, the prover cannot anticipate them, or be ready with answers simply by repeating those from a previous round of questioning.

Another key notion is that these proofs allow for the possibility of error, as long as it is a very small possibility. "If I am a liar, there is a small chance you will not catch me, but with overwhelming probability you will find an error," Goldwasser says. Relying on overwhelming probability rather than 100% certainty provides cryptographers with enough freedom to make these proofs useful, she explains. "In all of these proofs there is imperfection, and that's what allows this flexibility."

As it turns out, extensions of their theory of proofs provide a new way to classify difficult problems, which is important for the study of computational complexity.

When Micali and Goldwasser began their work in the early 1980s, today's Internet was a distant dream, and there were no cloud computing or mobile applications to make the need for data security as obvious as it is now. While their focus was largely on theory, it has become the basis for advanced cryptographic schemes that are starting to be introduced in practice, Rivest says.

"The necessity of using randomness in encryption was surprising to many people."

Bellare says the two laid the foundation for modern cryptography. "The field was created by their ideas 30 years ago," he says. It was not just their theoretical work, but their way of expressing their ideas, using such concepts as games and adversaries, that fired other researchers' imaginations. "They had a very strong sort of social and cultural impact, because they had a way of expressing ideas that was very catching," Bellare says.

The two do not yet know what they will do with the $250,000 prize that goes with the award, though Goldwasser says their children already have decided it should fund family trips to soccer matches in far-off countries.

Micali's advice for young computer scientists is to cultivate an initial sense of irreverence, a certain lack of respect for what has gone before. "You don't want to be constrained by current theories when you develop your own," he says. "Just go for what you like and be as irreverent as you can and then try to restore law and order afterwards."

Goldwasser says scientists should trust their own tastes, and take the time to really chew on a problem that interests them. "It's very useful to kind of try to think of the problem from lots of different directions," she says. Her other admonition: "Don't stop too early." Sometimes, she says, researchers will solve a problem, write it up for a conference, and move on without considering other aspects of their solution. "If you come up with something new, and it's yours, stick with it," she says.

Neil Savage is a science and technology writer based in Lowell, MA.

PHOTOGRAPHS BY BRYCE VICKMARK

Milestones | DOI:10.1145/2461256.2461282

ACM Honors Computing Innovators

ACM's awards celebrate achievements in networks, information retrieval, multi-agent systems, computer science education, versatile compiler technologies, and more.

ACM HAS ANNOUNCED the winners of several prestigious awards in recognition of contributions to computing technology. The 2012 ACM award winners include prominent computer scientists, educators, and entrepreneurs.

The Grace Murray Hopper Award, given to the outstanding young computer professional of the year, and its accompanying $35,000 honorarium, go to **Martin Casado** of VMware and Stanford University for his work in creating the movement of software-defined networking (SDN), and to **Dina Katabi** of the Massachusetts Institute of Technology (MIT) for her contributions to the theory and practice of network congestion control and bandwidth allocation.

The Paris Kanellakis Theory and Practice Award honoring specific theoretical accomplishments that have had a significant and demonstrable effect on the practice of computing, accompanied by a prize of $10,000, is being given to **Andrei Broder** of Google, **Moses Charikar** of Princeton University, and **Piotr Indyk** of MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) for their groundbreaking work on locality-sensitive hashing.

The Karl V. Karlstrom Outstanding Educator Award, accompanied by a prize of $5,000, goes to **Eric Roberts** of Stanford University for his contributions to computing education.

The ACM/AAAI Allen Newell Award is presented to an individual for career contributions that have breadth within computer science, or that bridge computer science and other disciplines. This endowed award, accompanied by a prize of $10,000 and supported by the Association for the Advancement of Artificial Intelligence, is being given this year to **Yoav Shoham** of Stanford University and **Moshe Tennenholtz** of Technion–Israel Institute of Technology and Microsoft Research Israel for their fundamental contributions at the intersection of computer science, game theory, and economics.

ACM's Software System Award is given to an institution or individual(s) in recognition of the development of a software system that has had a lasting influence, reflected in contributions to concepts, in commercial acceptance, or both. The 2012 Software System Award, which carries a prize of $35,000, is being awarded to **LLVM**, a language-independent collection of programming technologies that enables code analysis and transformation for arbitrary programming languages. Started in 2000 at the University of Illinois at Urbana-Champaign by **Chris Lattner** and **Vikram Adve**, LLVM has become widely used in both commercial products and for computer science research.

The Eugene L. Lawler Award for Humanitarian Contributions within Computer Science and Informatics recognizes an individual or a group who have made a significant contribution through the use of computing technology. This year the award, which includes a prize of $5,000 (plus travel expenses to the awards banquet), is being given to **Thomas Bartoschek** of the University of Münster and **Johannes Schöning** of Hasselt University for their contributions to GI@School (Geoinformatics at School), a program that encourages young people to develop a fascination for computer science and computer science research.

These innovators have made significant contributions that enable computer science to solve real-world challenges.

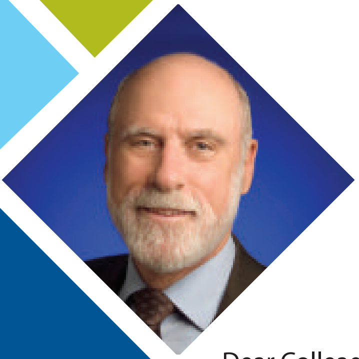
In addition, ACM will present its Distinguished Service Award to **Mateo Valero** for spearheading initiatives in Europe that have advanced high-performance computing research and education.

ACM has selected **Zvi Kedem** to receive its Outstanding Contribution to ACM Award for his leadership in rebuilding the ACM Computing Classification System (CCS) as a modern cognitive map of the computing field for the worldwide computing community.

Three ACM Presidential Awards are being given in recognition of leaders who extend ACM's profile and promote its role in advancing computing as a science and a profession. These honorees are:

- **Fabrizio Gagliardi** of Microsoft Research, for his leadership as chair of the ACM Europe Council.
- **Yunhao Liu**, professor at Tsinghua University, an active researcher and a member of the ACM China Council.
- **P J Narayanan**, professor and dean at IIIT, Hyderabad, and president of the ACM India Council.

ACM will present its awards at the ACM Awards Banquet on June 15 in San Francisco, CA. C



Association for Computing Machinery

Global Reach for Global Opportunities in Computing

Dear Colleague,

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

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

Timely access to relevant information

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

Resources to enhance your career

- **ACM Tech Packs**, exclusive annotated reading lists compiled by experts
- **Learning Center** books, courses, webinars and resources for lifelong learning
- Option to join **36 Special Interest Groups** (SIGs) and **hundreds of local chapters**
- **ACM Career & Job Center** for career-enhancing benefits
- ***CareerNews***, email digest
- **Recognition of achievement** through Fellows and Distinguished Member Programs

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

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

Sincerely,

Vint Cerf

President
Association for Computing Machinery

acm Association for Computing Machinery

Advancing Computing as a Science & Profession

Association for
Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: AD13

You can join ACM in several easy ways:

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

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

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

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

Please print clearly

Name

Address

City State/Province Postal code/Zip

Country E-mail address

Area code & Daytime phone Fax Member number, if applicable

Purposes of ACM

ACM is dedicated to:
1) advancing the art, science, engineering, and application of information technology
2) fostering the open interchange of information to serve both professionals and the public
3) promoting the highest professional and ethics standards

I agree with the Purposes of ACM:

Signature

ACM Code of Ethics:
http://www.acm.org/about/code-of-ethics

choose one membership option:

PROFESSIONAL MEMBERSHIP:

○ **ACM Professional Membership: $99 USD**

○ **ACM Professional Membership plus the ACM Digital Library: $198 USD ($99 dues + $99 DL)**

○ **ACM Digital Library: $99 USD (must be an ACM member)**

STUDENT MEMBERSHIP:

○ **ACM Student Membership: $19 USD**

○ **ACM Student Membership plus the ACM Digital Library: $42 USD**

○ **ACM Student Membership PLUS Print *CACM* Magazine: $42 USD**

○ **ACM Student Membership w/Digital Library PLUS Print *CACM* Magazine: $62 USD**

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

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

RETURN COMPLETED APPLICATION TO:

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

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

Satisfaction Guaranteed!

payment:

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

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

○ Professional Member Dues ($99 or $198) $ ____________________

○ ACM Digital Library ($99) $ ____________________

○ Student Member Dues ($19, $42, or $62) $ ____________________

Total Amount Due $ ____________________

Card # Expiration date

Signature

DOI:10.1145/2461256.2461266

Stas Filshtinskiy

Privacy and Security

Cybercrime, Cyberweapons, Cyber Wars: Is There Too Much of It in the Air?

Where reality stops and perception begins.

In the past the media focused on cyber criminals. For the last two years, whenever I see a news report related to unscrupulous developments in cyberspace, there is almost always a mention of weapons, the military, or an intelligence service. Nowadays, even when criminals are blamed for performing a major cyber heist, vendors call it "Operation Blitzkrieg" and the mass media announce, "Russian Hackers Declare War on USA." A *New York Times* article attributed an attack by "Izz ad-Din al-Qassam Cyber Fighters" against U.S. banks to the state of Iran without any evidence other than "a level of sophistication far beyond that of criminals."[2]

Has the world really changed that much in two years? I don't think so. Even the most complex cyber attacks are within the reach of cyber criminal enterprises.

Criminals have always raced ahead of the pack, figuring out how to steal from somewhere before the rest of the population realized there was money to be had. Cyber criminals have sites where they sell and buy things. In the early 2000s, criminals were selling credit card numbers.[6] Then banks went online, and criminals invented phishing. As losses grew, the financial institutions responded by improving their security technologies. But cybercrime had already moved on to the next best fraud.

Even the most complex cyber attacks are within the reach of cyber criminal enterprises.

Criminals are open-minded when it comes to new ways of stealing money. They learn fast. The biggest change in the business of cybercrime occurred when the most advanced groups moved from selling goods (stolen data or computer viruses) to the establishment of the criminal cyber services (stealing data, providing access to infected computers, or writing tools to steal data).

This transition in criminal business models was good for risk-averse cybercriminals.[3,4] It gave them stable cash flow and reduced their risks. It allowed them to interact with their customers (other criminals) without ever getting physically close to them. This approach attracted much less attention from law enforcement and old-style criminals—those carrying guns instead of laptops. Computer crime became an industry comparable in size to weapons trafficking and drug trafficking. Various sources put individual monetary losses from cybercrime as more than $100 billion. Symantec in the 2012 Norton Cyber-

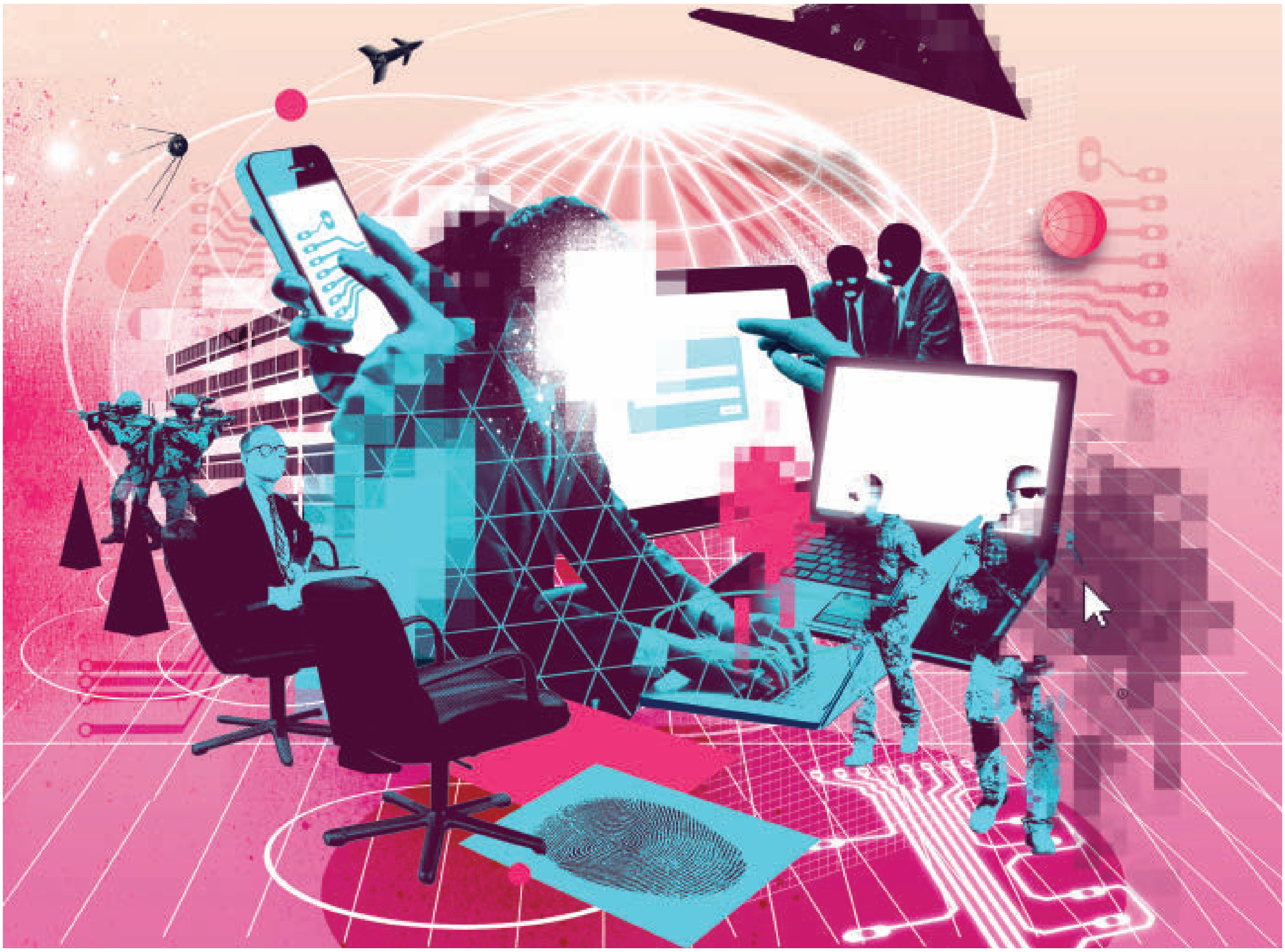


ILLUSTRATION BY MARIO WAGNER

crime report estimated an annual cost of up to $110 billion.[8] Such reports might or might not be accurate, but even 1% of the perceived losses is a lot of money.

How can such money be made? Garden-variety criminals cannot pull off such expensive heists. That money comes from sophisticated, interlinked services that criminals have on offer. Here the some of the services available on cybercriminal trade portals:

- Sending unsolicited messages of all sorts—this now includes not only email messages, but also Twitter and social network messaging.
- Writing malware on-order, which includes online support and regular updates for additional licensing fees.
- Bulletproof—or as it is often termed, "abuse resistant"—hosting, for those criminals who need to have Web presence.
- Botnet access.
- Anonymous access to the Internet.
- Getting your video to the top of YouTube.
- Hacking in general.

These services are on the market for anyone who wants to buy them—governments, activists of all persuasions, terrorists, and criminals. These services facilitate other criminal activities and are available for anyone who can pay. According to an interview with a provider,[1] a denial-of-service attack is priced between $50 and $500 per day,[a] depending on the site and deployed defenses. This provider estimated the price of shutting down the popular blogging site LiveJournal.com at $250 to $400 per day.

Criminals have advertised:

- The price for hacking a private email address is between $30 and $50.
- A forged copy of an identity document of virtually any country in the world costs less than $30.
- Custom-made software to automatically register new accounts on popular Web sites and bypass CAPTCHA protection costs less than $500.
- Custom-built malware costs $1,500 plus monthly support and consultation fees.

Cybercrime services allow businesses (for example, street gangs with soldiers on the ground) to buy a supply line of stolen credit card data or bank credentials belonging to individuals or companies local to their area. Once they pay for the service, these "businesses" can exploit this information at their own risk. The suppliers are not there if the exploiters of the data are caught. They are jurisdictionally and logically far away from the crime and out of law enforcement's way. Successful arrests of providers of cybercriminal services are rare and require a long-term sting operation or entrapment like Operation Card Shop, which was a two-year undercover effort by the FBI that concluded in mid-2012.

Cyber criminals' capabilities are impressive. Now consider some attacks that have been attributed to intelligence services, often with language about cyberweapons. According to media reports, the proverbial

[a] All prices are in U.S. dollars

crown jewels of the well-known security vendor RSA were stolen and allegedly used to attack multiple targets, including financial organizations and weapons manufacturers. The attack was not very advanced—it started with a known exploit, continued for some time, and ended with exfiltration of the data through a typical channel.

The Stuxnet attack occurred when a uranium enrichment plant in the Islamic Republic of Iran was sabotaged. The attack allegedly used specially crafted malware, delivered to the target by uncontrolled USB devices. The attack exploited previously known and unknown vulnerabilities in industrial control systems to damage centrifuges.

Georgia, a small nation in the Caucasus Mountains, got into the bad books of its bigger neighbor Russia over the future of two pro-Russian separatist regions. It resulted in military conflict. Separatists' online news agencies were allegedly compromised by hackers associated with Georgia while the online capabilities of Georgia were severely degraded by a massive denial-of-service attack. Georgian official and private websites were also defaced.

An attack sponsor need not be a hacker or social engineer to profit from the theft of valuable data.

The main shared feature of each of these stories is that those attacks used nothing more than was available in the criminal markets at the time. Some of the example attacks may have been the work of government agencies,[b] but they are also within reach of determined criminal groups. Similar attacks can be easily designed from building blocks available on the market. Sophisticated malware can be ordered online. Unknown (so-called zero-day) vulnerabilities can be purchased and turned into exploits. Computing power equivalent of multiple, top-of-the-range supercomputers is on offer. Databases of already-hacked passwords are available.

An attack sponsor need not be a hacker or social engineer to profit from the theft of valuable data. A decent project manager capable of understanding what items are in demand can identify particular information as marketable and build and execute a project plan using purchased components and services. Custom exploits can deliver the payload into a protected perimeter, unique malware can search and eventually reach valuable data, and individually crafted software can exfiltrate the loot. The sponsor of the attack can then sell the data wholesale or piece by piece to any party able to pay, whether a criminal organization, intelligence service, or terrorists. The scariest thing of all is that most of these recent attacks could be the work of a criminal.

According to security vendors, policymakers, and media, the world is rife with secret services, intelligence operatives, and military commands engaged in cybercrime. This perception is partially based on truth: intelligence agencies and military do operate in cyberspace. But this perception leads to bad decisions. Business leaders are not sure how best to invest in protection. Political leaders pass laws that reduce freedom of information on the Internet and empower counterintelligence services. Society is exposed because defenses appropriate to the threat are not built.

Most attacks, regardless of who is paying for them, are perpetrated by cyber criminals. We need to oppose them through better international enforcement efforts, even though this has been difficult to achieve. We must also disrupt their business models by taking down their ability to offer and deliver their services. This has been done somewhat successfully by U.K. banks.[5] Most important, we must recognize that most attacks are executed by criminal enterprises, and not by nation-states. These attacks can be defended against if we put in the tools to do so.

[b] The Stuxnet attack was attributed to the U.S. government according to David Sanger.[7]

References

1. AiF 2011 in Russian. Interview with the hacker about DDoS attacks; http://www.aif.ru/techno/article/42414.
2. Bank hacking was the work of Iranians, officials say. *The New York Times* (Jan. 8, 2013); http://www.nytimes.com/2013/01/09/technology/online-banking-attacks-were-work-of-iran-us-officials-say.html?_r=0&pagewanted=print
3. Chiesa, R.N. Cybercrime and underground economy: Operating and business model; http://www.flarenetwork.org/report/enquiries/article/cybercrime_and_underground_economy_operating_and_business_model.htm.
4. Filshtinskiy, S. Cyber criminal economy. In *Proceedings of the AusCERT 2007 Conference*.
5. Financial Fraud Action U.K. 2011. Fraud, the facts; http://www.financialfraudaction.org.uk/Publications/#/52/.
6. Kabay, M.E. A brief history of computer crime; http://www.mekabay.com/overviews/history.pdf
7. Sanger, D. *Confront and Conceal*. Crown Publishers, 2012, 197–203.
8. Symantec. 2012 Norton cybercrime report. 2012 Norton Cybercrime Report. http://now-static.norton.com/now/en/pu/images/Promotions/2012/cybercrimeReport/2012_Norton_Cybercrime_Report_Master_FINAL_050912.pdf.

Stas Filshtinskiy (Stas.Filshtinskiy@baesystemsdetica.com) is a principal consultant with BAE Systems Detica in Melbourne, Australia.

The views expressed in this column are those of the author and do not necessarily represent the views of the author's employer or any organization with whom the author might be associated.

DOI:10.1145/2461256.2461267

Phillip G. Armour

The Business of Software
What Is a "Good" Estimate?

Whether forecasting is valuable.

Our measure of the "goodness" of an estimate is usually based on one thing: how closely the forecast ends up matching what we actually see. The search for the "accurate estimate" is one of the El Dorado quests of software project management. Despite the clearly oxymoronic nature of the phrase,[a] the most common question I am asked about an estimate is "...how accurate is it?" It seems to be a very natural question and one we might ask of a car mechanic or a builder. However, "accuracy" is only one yardstick we could use to assess how good an estimate is—there are other criteria and the meaning of "accurate" could bear some scrutiny. But first, the weather.

ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

Murphy's Lore

When considering weather prediction, the late Allan Murphy of Oregon State University suggested three attributes of a forecast that determine its "goodness."[1] With a little adjustment, we can apply Murphy's principles to estimating software development projects.

The three attributes of "goodness" Murphy noted were:

- Consistency
- Quality
- Value

To these three attributes, we can add three more: honesty, accuracy, and return. These three additions are closely aligned with Murphy's attributes so we can pair them together.

Consistency and Honesty

To be consistent, the process used to create an estimate must be rational (for example, no random guesses or wishes) and grounded in some knowledge base of relevant data. Ideally this knowledge base would also be consistent, representing the performance of similar projects, but this is not mandatory. If identical project data is not available then using history of somewhat dissimilar projects simply introduces additional uncertainty in the estimate output. As long as the uncertainty is honestly calculated and openly presented, it just becomes one of the considerations in making any business decision based on the estimate.

Also included in this category is the requirement that all the data pertinent to the project be included in the estimate. Data cannot be cherry-picked to achieve a desired result. This should especially include the inherent uncertainty in whatever result is forecast.

An estimate is "honest" if it truly reflects the best judgment of the estimator and the true output of the rational process used. I have often heard project managers complain they experienced intense pressure to lower their best-judgment estimates if they did not fit within preconceived or predefined commitment levels. Sometimes estimators anticipate this and proactively lower their estimates to make them more acceptable to the decision makers. Such pressures may be

[a] As pointed out in P.G. Armour, "The Inaccurate Conception." *Commun. ACM 51*, 3 (Mar. 2008).

overt or covert but they almost always work to make the estimate less honest and less valuable.

Quality and Accuracy

This is the most common measure by which estimates are judged, but we need to examine what we mean by "accuracy." The usual interpretation is the degree of correspondence between the estimate and the actual result. However, at the time of estimating, there is no way to assess quality using this definition. We do not have the actual result in advance of running the project, so an estimate is accurate compared to…what?

Estimates—of rainfall or projects—are intrinsically probabilistic. Simply because a result does not exactly correlate to an estimate does not mean the estimate was wrong or even that it was inaccurate.[b] An estimate is inaccurate if its probability of result does not match the probability of the actual. If a project is estimated as having a 20% likelihood of success and it "fails" by overrunning the budget or the schedule, perhaps the project was flawed but arguably the estimate was accurate since it forecast a high probability of failure.

If we adjust our definition of the word "accurate" there are other valuable comparisons we can make. For instance, at the time we estimate, we can assess its quality and "accuracy" by comparing the estimate against the knowledge base mentioned earlier. If an estimate falls outside of the normal range of variability of similar projects it is reasonable to assert it is inaccurate.

The most obvious way to assess estimation accuracy is a posteriori, once we have the actual results. To do this correctly or, well, accurately would require reconstructing the estimate while accounting for the variance in data that was observed. When we do that, however, we are actually assessing the viability of the process and data used in the original estimate. For example, if a project was completely de-staffed for a while to deal with some unexpected emergency in the organization, it is unlikely the original estimate would correlate to the result. If, when the project finished (or during the destaffing period) the estimate was rerun incorporating this new information, perhaps it would correlate well and could be considered "accurate."

Murphy notes a host of sub-attributes of "quality" as statistical assessments of probability distributions. Assessing these usually requires multiple forecasts—something that is common in meteorology but not in software. We can and should reestimate as data becomes more available. The data this would provide would allow us to apply statistical quality measurements to our estimates and estimation process—but first we would have to estimate more than once.

Value and Return

This aspect of estimation goodness is often totally ignored. The act of estimating is the purchasing of information. We expend a certain amount of time and effort to obtain some knowledge about a project. This effort costs money and the knowledge has worth. A "good" estimate maximizes the return on this investment by obtaining the highest value at the lowest cost.

Estimates have no intrinsic value; their value is determined entirely by how they are used to make business decisions. The consequences of those decisions might be quite unrelated to the technical aspects of producing a project forecast. A quick and rough estimate used to cancel an infeasible project might be a lot more valuable than a time-consuming and expensive estimate used to justify a marginal project. The true value of an estimate is realized mostly by the difference in the business decisions made.

There are four aspects to the value of estimates[3,4]: (a) What business choices might be indicated by the estimation output? (b) What are the benefits/costs of each choice? (c) What is the quality of the information available to make the decision without the estimate? (d) What is the quality of the estimate? This is a complex subject and quite situation-specific, but factors (a) and (b) are clearly business issues independent of any estimate. Factors (c) and (d) represent the incremental benefits of making decisions based on the defined process rather than whatever other approach (such as guessing or wishing) might be used. The value component of an estimate's goodness is not under the control of the estimator but is essential to providing a justifiable return and it might be the most important attribute of an estimate.

Getting Good

To improve the "goodness" of estimates we must address all of these factors: 1. Understand how the estimate output will be used and how it will guide the business choices. 2. Perform a tradeoff analysis to determine the most optimal (and achievable) result that can be obtained.[c] 3. Use a consistent process based on the most relevant historical data available. 4. Assess the correlation of the estimate to its knowledge base and present it in a useful way. 5. Require an honest expression of the estimator's judgment independent of bias and pressure. 6. Express the estimate output in probabilistic terms. 7. Track project performance against the forecast and adjust for variance. 8. When projects finish, recalibrate the basis of estimates for next time.

Projects and Weather

Both are complex systems with a lot of interacting factors; both are somewhat nondeterministic but have trends that, probabilistically, can be measured. More importantly, both have measurements and forecasts that can be very valuable. We may complain about the "accuracy" of weather forecasts but, by applying honest and rational processes over the last 30 years, they have been steadily improving and becoming more valuable.[2]

We could do the same in software development. C

[b] P.G. Armour, "The Inaccurate Conception." *Commun. ACM 51*, 3 (Mar. 2008).

[c] See P.G. Armour, "The Goldilocks Estimate." *Commun. ACM 55*, 10 (Oct. 2012).

References

1. Murphy, A.H. What is a good forecast? An essay on the nature of goodness in weather forecasting. *American Meteorological Society Journal 8*, 2 (June 1993), 281–293.
2. Silver, N. *The Signal and the Noise: Why So Many Predictions Fail—But Some Don't*. Penguin Press, 2012, 126.
3. Sonka, S.T. et al. Economic use of weather and climate information: Concepts and an agricultural example *J. Climatology 6* (1986), 447–457.
4. Winkler, R.L., Murphy A.H., and Katz, R.W. The value of climate information: A decision-analytic approach. *J. Climatology 3* (1983), 187–197

Phillip G. Armour (armour@corvusintl.com) is a senior consultant at Corvus International Inc., Deer Park, IL, and a consultant at QSM Inc., McLean, VA.

DOI:10.1145/2461256.2461268

George V. Neville-Neil

Article development led by acmqueue
queue.acm.org

Kode Vicious
Swamped by Automation

Whenever someone asks you to trust them, don't.

Dear KV,

As part of a recent push to automate everything from test builds to documentation updates, my group—at the request of one of our development teams—deployed a job-scheduling system. The idea behind the deployment is that anyone should be able to set up a periodic job to run in order to do some work that takes a long time, but that is not absolutely critical to the day-to-day work of the company. It is a way of avoiding having people run `cron` jobs on their desktops and of providing a centralized set of background processing services.

There are a couple of problems with the system, though. The first is that it is very resource intensive, particularly in terms of memory use. The second is that no one in our group knows how it works, or how it is really used, but only how to deploy it on our servers—every week or so someone uses the system in a new and unexpected way, which then breaks the system for all the previous users. The people who use the system are too busy to explain how they use it, which actually defeats the main reason we deployed it in the first place—to save them time. The documentation is not very good either. No one in the group that supports the system has the time to read and understand the source code, but we have to do something to learn how the system works and how it scales in order to save ourselves from this code. Can you shed some light on how to proceed without becoming mired in the code?

Swamped

Dear Swamped,

So your group fell for the "just install this software and things will be great" ploy. It is an old trick that continues to snag sysadmins and others who have supporting roles around developers. Whenever someone asks you to trust them, don't. Cynical as that might be, it is better than being suckered.

But now that you have been suckered, how do you un-sucker yourselves? While wading through thousands of lines of unknown code of dubious provenance is the normal approach to such a problem—a sort of "suck it up" effort—there are some other ways of trying to understand the system without starting from `main()` and reading every function.

The first is to build a second system, just for yourselves, and create a set of typical test jobs for your environment. The second is to use the system already in place to test how far you can push it. In both cases, you will want to instrument the machine so that you can measure the effect that adding work has on the system.

PHOTOGRAPH BY IGOR TEREKHOV

Once you have the set of test jobs or you are running on the production machine, you instrument your machine(s) to measure the effect each job has on the system. In your original question, you say memory is one of the things the job-control system uses in large amounts, so that is the first thing to look at. How much real memory, not virtual, does the system use when you add a job. If you add two jobs, does it take twice as much? What about three? How does the memory usage scale? Once you can graph how the memory usage scales, you can get an idea of how much work the system can take before you start to have memory problems. You should continue to add work until the system begins to swap, at which point you will know the memory limit of the system.

Do not make the mistake of trying only one or two jobs—go all the way to the limit of the system, because there are effects you will not find with only a small amount of work. If the system had failed with one or two jobs, you would not have deployed it at all, right? Please tell me that is right.

Another thing to measure is what happens when a job ends. Does the memory get freed? On most modern systems you will not see memory freed until another program needs memory, so you will have to test by running jobs until the system swaps, then remove all the jobs, and then add the same number of jobs again. Does the system swap with fewer jobs after the warm-up run? The system may have a memory leak. If you cannot fix the leak, then guess what, you will get to reboot the system periodically, since you are unlikely to have time to find the leak yourself.

When you are trying to understand how a system scales, it is also good to look at how it uses resources other than memory. All systems have simple tools to look at CPU utilization, and you should, of course, make sure that the job-control system is the one taking all the CPU time, as that adds to the total system overhead.

When you are trying to understand how a system scales, it is also good to look at how it uses resources other than memory.

The files and network resources a system uses can be understood using programs such as `netstat` and `procstat`, as well as `lsof`. Does the system open lots of files and just leave them? That is a waste of resources you need to know about, because most operating systems limit the number of open files a process can have. Is the system disk intensive, or does it use lock files for a lot of work? A system that uses lots of lock files needs to have space on a local, non-networked disk for the lock files, as network file systems are particularly bad at file locking.

A rather drastic measure, and one that I favor, is the use of `ktrace`, `strace`, and particularly `DTrace` to figure out just what a program is doing to a system. The first two will definitely slow down the system they are measuring, but they can quickly show you what a program is doing, including the system calls it makes when waiting for I/O to complete, plus what files it is using, and other details. On systems that support `DTrace`, the overhead of tracing is reduced, and on a system that is not latency sensitive, it is acceptable to do a great deal more tracing with `DTrace` than with either `ktrace` or `strace`. There is even a script, `dtruss`, provided with `DTrace`, that works like `ktrace` or `strace`, but that has the lower overhead associated with `DTrace`. If you want to know what a program is doing without tiptoeing through the source code, I strongly recommend using some form of tracing.

In the end it is always better to understand the goals of a system, but with engineers and programmers being who they are, this might be like pulling teeth. Not that pulling teeth isn't fun—trust me, I've done it—but it is more work than it looks like and sometimes the tooth fairy doesn't give you that extra buck for all your hard work.

KV

Related articles on queue.acm.org

Scale Failure
George Neville-Neil
http://queue.acm.org/detail.cfm?id=2147781

Orchestrating an Automated Test Lab
Michael Donat
http://queue.acm.org/detail.cfm?id=1046946

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

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

PHOTOGRAPH BY IGOR TEREKHOV

DOI:10.1145/2461256.2461269

Peter J. Denning

The Profession of IT
Thumb Numbers

Rules of thumb stated as numerical rules are enticing, but many are folk theorems that may not apply in your critical situation.

RULES OF THUMB are common and often very helpful. They convey bits of wisdom accumulated by many people over a long time. Parkinson's Law warns that things seldom get done early: "Work expands to fill the time available for its completion." Murphy's Law warns against complacency: "Whatever can go wrong, will." Hofstadter's Law captures the project planner's conundrum: "Projects always take twice as long as planned, even when this law is taken into account." Even the self-contradictory adage, "The exception proves the rule," is taken as rule of thumb.

Rules of thumb seem much more concrete, authoritative, and universal when they contain numbers. I have picked three examples to examine here. Despite their popularity, they are very shaky as general rules:

- 80-20 Rule: 80% of production comes from 20% of producers.
- 7 Chunks Rule: Our mental span of control is limited to about 7 chunks.
- 7% Contents Rule: The credibility of your message depends 93% on your tone of voice and body language, and only 7% on the content of your words.

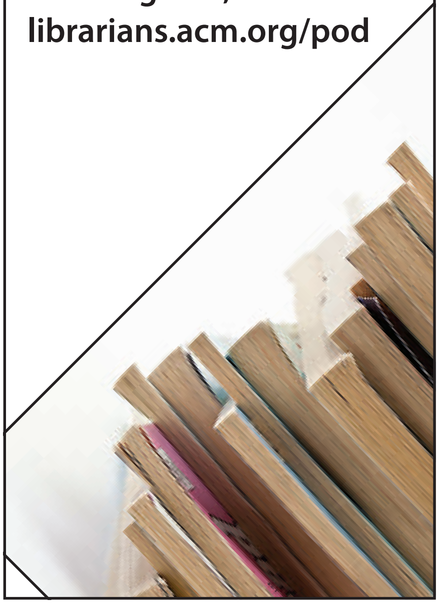
Even as heuristics, these "rules" are not very reliable. We can only have confidence that a rule applies when we have data to ground our conclusions.

PHOTOGRAPH BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

The 80-20 Rule

The 80-20 rule is a statement about a population of unequal producers. It says if you rank order the members from largest to smallest productivity, you will find that 80% of the production comes from the first 20% of the ranking. The rule is named after the Italian economist Vilfredo Pareto. In 1906 he observed that 80% of the land in Italy was owned by 20% of the population. He also observed in his garden that 80% of the peas came from 20% of the pods. The Pareto effect has been observed in many other cases and has led to statements of the following kinds in many fields:

- About 80% of the world's GDP is produced by 20% of the countries (United Nations Development Program, 1992).
- About 80% of your profits (or complaints!) come from 20% of your customers.
- About 80% of injuries come from 20% of known hazards.
- About 80% of crimes are committed by 20% of criminals.
- About 80% of the vulnerabilities in a critical infrastructure reside in 20% of the nodes.
- About 80% of bug reports will be eliminated by fixing the top 20% of known bugs (Microsoft Security Development Lifecycle).
- About 80% of Internet traffic goes to 20% of the nodes.



- About 80% of your software specifications can be implemented with only 20% of the full-project effort (the rapid prototyper's motto).

There are many more examples using different ratios. For example, "You will be most successful at weight loss if 90% of your foods are healthy and 10% are 'fun' foods." And the ever popular, "About 99% of the wealth is held by 1% of the taxpayers." Without much searching, you will easily find many more examples like these.

Some of these claims are validated by data, but many are simply asserted based on an assumption that the Pareto principle is universal. Is it?

To answer that question, let's go back to basics. We are given a set of producers. Associated with each producer x is $a(x)$, the amount that x has produced. The measure $a(x)$ can also be the number of occurrences (frequency) of x, because x is "producing" occurrences. The producers have been ranked (labeled) in order of decreasing production. If A is the total amount from everyone, we can set the proportion of production from x as $p(x) = a(x)/A$. Statisticians define these distributions as "Pareto distributions" when $p(x)$ decays proportional to x^{-a}, where a is a parameter.

In the special case a=1, the frequency (production) of x is proportional to 1/x and the distribution is called Zipf's Law. Zipf's Law is named after George Kingsley Zipf (1902–1950), who observed that in compilations of words from various languages sorted by decreasing frequency, the frequency of any word tends to be inversely proportional to its rank.

Another common case is a=2, in which case the distribution is often called a power law (or more precisely "inverse square power law"). In this case, doubling the rank cuts production to a quarter. An example is Internet connectivity, where the $p(x)$ is the relative number of nodes that have x connections to other nodes.[2]

The accompanying figure shows two data sets plotted on log-log scales. When the data on a log-log graph fall on a straight line of slope -a, they obey a power law with parameter a. The upper line plots the data of a Zipf Law (a=1) and the lower line a power law (with a=2). In other words, we can confirm how closely our data follow a Pareto distribution by plotting them on a log-log graph and seeing how closely they follow a straight line.

The 80-20 cutoff point is not a general feature of Pareto distributions. In the data for the figure, the Zipf-Law data displayed an 80-54 cutoff, and the power-law data displayed an 80-4 cutoff. Only when a=1.32 did the 80-20 rule work exactly. For continuous data, a=1.16 makes it work (Wikipedia).

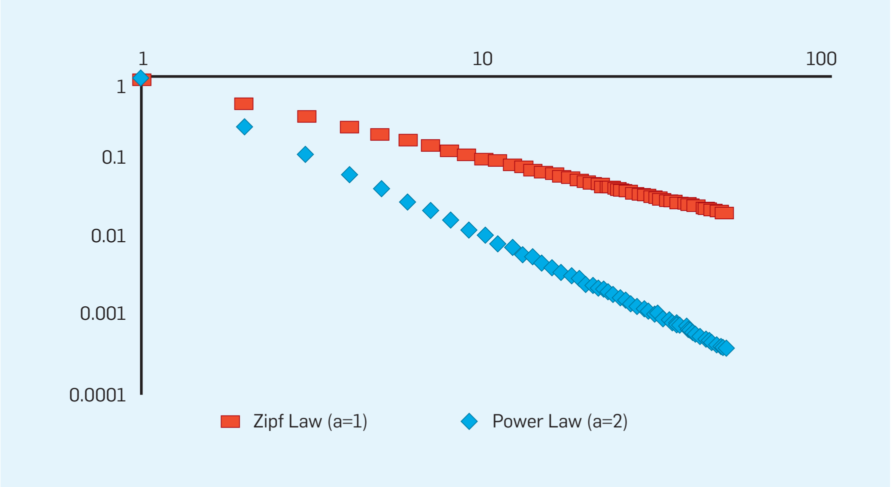
So the "80-20 rule" is little more than a folk theorem based on a few prominent cases. It holds only for a small subset of Pareto distributions. It is not a universal law. Thus, for instance, network scientists who believe that disabling a handful of highly connected "hub" routers could shut down the Internet are mistaken because the real Internet is engineered for more redundancy.[1] Similarly, rapid prototypers who believe that 80% of the specifications can be completed with 20% of the total project effort are on shaky ground.

When good data are available, it is easy to calculate a cutoff point and put it to good use. For instance, an engineer might have discovered that 10% of the electric-grid nodes account for 90% of the vulnerabilities; hardening those nodes is a good use of limited infrastructure protection funds.

While we may have trouble predicting where inequality of production arises or what causes it, we can be sure that almost all the time production will be unequal. Enterprise software expert Rick Hayes-Roth told me of a practical way, inspired by Alan Lakein,[3] to exploit inequality for software development. Make a list of all the outcomes

People's memories are more powerful when they incorporate more meaning, relationships, and context.

Example log-log plots of two Pareto datasets.



your software is to produce. Rank each one as priority A (top), B (middle), and C (low) with the constraints that at least 1/3 be rated C and at most 1/3 be rated A. Then ignore B and C tasks and do only A tasks. Customer groups can produce these rankings by giving everyone enough votes to cover 1/3 of the list of possible outcomes. Hayes-Roth says that when time pressures increase (as when technology accelerates), the rewards of this approach increase. Getting the A priorities delivered on time will keep you alive, while chasing after B's and C's will sink you.

The 7 Chunks Rule

In 1956 psychologist George Miller published in *Psychology Review* a study called "The magical number seven, plus or minus two," where he found that most people can remember between five and nine "chunks" of information in their short-term memories.

Miller's paper became very popular. It led to popular notions such as a manager should manage about seven direct reports, and more than that become unmanageable. The idea that management span of control is ideally around seven has been taken as a law even though there is little data to support it. Some managers have shown great skill with many more than seven reports, and others have trouble with two or three.

Later studies of human memory have shown that people can learn to remember many more than seven items after being trained in memorization methods. They form hierarchies grouping chunks at successive levels, all linked together by stories and substories. The real lesson is not that short-term memory is a limitation, but that people's memories are more powerful when they incorporate more meaning, relationships, and context.

The 7% Content Rule

In 1971, Albert Mehrabian of the UCLA business school published a book about factors that lead customers to like or dislike a salesperson.[4] He concluded that the words of the sales message account for 7% of the "like" assessments, tone of voice for 38%, and body language for 55%. People are especially sensitive to incongruities, for example someone claiming to have no complaint about you while avoiding eye contact with you. In that case, listeners tend to go with their sense of voice and body language rather than the content of the words spoken.

Many people have seized on this study as proof that nonverbal communication is more important than verbal. Allen Weiner has written a book about how you can conduct yourself in the workplace by cultivating good practices of voice and body language.[5] Although supposedly derived from this rule, most of Weiner's excellent advice does not depend on the truth of a 7-38-55 rule.

There are many reasons to doubt the universality of this rule. Mehrabian wrote about listeners reacting to recordings of single words, rating the emotional content of the words, and seeing if the rated emotions agreed with facial photographs of speakers. Mehrabian himself emphasizes the studies were about communicators talking about their feelings or attitudes and that little can be inferred for other contexts.

Philip Yaffe attacks this rule.[6] Many communications are in the form of speech or text; they are delivered by email or Web pages, not good media for communicating voice or body language. In delivered speech or a conversation, the persuasiveness is mostly a function of the words and their resonance with the concerns of the listener. Certainly, incongruous voice or body language can undermine your listener's trust, but no amount of those factors will overcome the lack of content. Yaffe points out that Abraham Lincoln's Gettysburg Address is one of the most famous speeches of all time, and yet no one has the slightest idea of Lincoln's tone of voice or body language.

Conclusion

There are many simple, easy-to-remember numerical rules of thumb. Many are catchy and seem to accord with our experience. They become "sticky memes" that people pass around as conventional wisdom. Not suspecting these sticky stories are mostly anecdotal, many people draw unwarranted conclusions.

When we teach math and computing, we know it is a serious mistake to start with the simple mathematical law abstracted from generations of thought about many real cases. It is far better to teach meaningful examples, and then summarize them with a law.

As a rule of thumb, rules of thumb are most useful when they summarize well-understood real-world experience. In the hands of the inexperienced, they are easily misapplied.

References

1. Alderson, D. In search of the real network science. *ACM Ubiquity* (Aug. 2009); http://ubiquity.acm.org/article.cfm?id=1595423.
2. Barabasi, A.-L. *Linked*. Plume, 2003.
3. Lakein, A. *How to Get Control of Your Time and Your Life*. Signet, 1989.
4. Mehrabian, A. *Silent Messages*. Wadsworth, 1971.
5. Weiner, A. *So Smart But ... : How Intelligent People Lose Credibility—and How They Can Get It Back*. Wiley, 2007.
6. Yaffe, P. The 7% Rule: Fact, fiction, or misunderstanding. *ACM Ubiquity* (Oct. 2011); http://ubiquity.acm.org/article.cfm?id=2043156.

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

DOI:10.1145/2461256.2461270

Nancy G. Leveson

Inside Risks

Learning from the Past to Face the Risks of Today

The Space Shuttle software program can provide guidance to today's projects.

As software takes over more and more functions in our increasingly complex and potentially dangerous systems, our software engineering problems are going to increase. The number of failures of large system projects we have been experiencing, particularly government systems, for example,[1–4] is not going to be acceptable. We need to learn from the failures and—even more important—from the successes of the past.

I recently contributed a chapter on software for a forthcoming book on the legacy of the Space Shuttle. A mythology has arisen about the Shuttle software with claims being made about it being "perfect software" and "bug free" or having "zero defects," all of which are untrue. But the overblown claims should not take away from the remarkable achievement by those at NASA and its major contractors (Rockwell, IBM, Rocketdyne, Lockheed Martin, and Draper Labs) and smaller companies such as Intermetrics (later Ares), who put their hearts into a feat that required overcoming what were tremendous technical challenges at that time. They did it using discipline, professionalism, and top-flight management and engineering skills and practices too often missing from today's safety-critical software projects. Much can be learned from this effort that is still applicable to the task of engineering complex software today. This column summarizes some of these lessons.

The Space Shuttle Atlantis during the STS-71 mission in 1995.

PHOTOGRAPH COURTESY OF NASA

There can always be differing explanations for success (or failure) and varying emphasis can be placed on the relative importance of the factors involved. Personal biases and experiences are difficult to remove from such an evaluation. But most observers agree that the process and the culture were important factors in the success of the Space Shuttle software as well as the strong oversight, involvement, and control by NASA.

Strict Government Oversight and Learning from the Past

The Shuttle software project benefited from what NASA had learned from earlier spacecraft. Gemini (1965–1966) was the first U.S. manned spacecraft to have a computer onboard. At the time, computer programming was considered an almost incidental activity. Experts wrote the software in low-level, machine-specific assembly languages. Fortran was considered too inefficient for use on real-time systems: The Gemini software development was largely haphazard, undocumented, and highly idiosyncratic.[7]

Computers had little memory at the time and squeezing the desired functions into the available memory became a difficult exercise and placed limits on what could be accomplished. The programmers also discovered that parts of the software were unchanged from mission to mission. To deal with these challenges, the designers introduced modularization of the code by loading only the functions required at that point in time. Another lesson learned was the need for software specifications and simulation programs to validate the guidance equations.

Despite the challenges and the low level of software technology at the time, the Gemini software proved to be highly reliable and useful. NASA realized, however, that the handcrafted, low-level machine code of Gemini would not scale to the complexity of later spacecraft. The problem of how to generate reliable and safe software had to be solved.

NASA used the lessons learned from the Gemini project about modularity, specification, verification, and simulation in producing the more complex Apollo software. In turn, many of the lessons learned from Apollo were the basis for the successful procedures used on the Shuttle.

Computers had little memory at the time and fitting necessary functions into the Apollo computer memory resulted in the abandonment of some features and functions and resulted in the use of tricky programming techniques to save others. The complexity of the resulting code led to difficulty in debugging and verification and therefore to delays. When it appeared that the software would be late, more people were added to the software development process, which simply slowed down the project even more. This basic principle that adding more people to a late project makes it even later is well known now, but it was part of the learning process at that time. Configuration control software was also late, leading to delays in supporting discrepancy reporting.

Another critical mistake, still made too often today, was to take shortcuts in testing when the project started to fall behind schedule. The 1967 Apollo launchpad fire gave everyone time to catch up and fix the software, as later the Challenger and Columbia accidents would for the Shuttle software. The time delay allowed for significant improvements in the software and in the process. Without it, the results may not have been as good.

One of the most important lessons was that software is more difficult to develop than hardware.

To reduce communication problems and control the development process, NASA created a set of control boards that managed all onboard software changes for Apollo. NASA also created a set of reviews for specific points in the development process, now familiar for many government or large company projects today. The review and acceptance process provided for consistent evaluation of the software and controlled changes, which helped to ensure high reliability and inserted much-needed discipline into the software development process. This control board and review structure became much more extensive for the Shuttle.

In the process of constructing and delivering the Apollo software, both NASA and the MIT Instrumentation Lab (which created the software) learned a lot about the principles of software engineering for real-time systems and gained important experience in managing a large real-time software project. These lessons were applied to the Shuttle. One of the most important lessons was that software is more difficult to develop than hardware. As a result:

- Software documentation is crucial.
- Verification must be thorough and proceed through a sequence of steps without skipping any or being rushed to try to save time.
- Requirements must be clearly defined and carefully managed before coding begins and as changes are needed. The dynamic nature of requirements for spacecraft should not be used as an excuse for poor quality.
- Good development plans should be created and followed.
- Experienced personnel should be assigned to a project early, rather than using the start of a project for training inexperienced personnel.
- Software should not be declared complete in order to meet schedules, requiring users to work around errors. Instead, quality should be the primary consideration.

NASA also learned three general and critical lessons: that increased attention to software would be necessary in future manned space programs; software needs the same type of disciplined and rigorous processes used in other engineering disciplines; and quality must be built in from the beginning—quality cannot be added after the software is written.

Using these lessons learned, the software development for Skylab followed strict engineering principles, which were starting to be created at that time in order to change software development from a craft to an engineering discipline. The Skylab program demonstrated that careful management of software development, including strict control of changes, extensive and preplanned verification activities, and the use of adequate development tools, results in high-quality software with high reliability.

Slowly and carefully NASA learned how to develop more complex software for spacecraft. The increasing success was not due simply to individuals learning from their mistakes, but the organization itself identifying the problems and ensuring the successful solutions derived from them were implemented and improved in later projects. Basically, NASA engaged in successful organizational learning.

To ensure these lessons would be applied in the Shuttle software and they would not have to relearn the same lessons from scratch for each spacecraft project, NASA maintained direct control of the Shuttle software rather than ceding control to the Shuttle hardware contractor. The hardware and software contracts were separated, with the software contractors directly accountable to NASA management. NASA had learned how important software was to the success of the entire program and closely managed the contractors and their development methods.

NASA worked very closely with the contractors and even constructed their own software development "factory" (the Software Production Facility) and test facility (SAIL) at NASA Houston, thus ensuring the highest standards and processes available at the time were used and that every change to human-rated flight software during the long life of the Shuttle was implemented with the same professional attention to detail.

The level of participation and control exercised by NASA is unusual for most government projects today, including many current NASA projects, where privatizing is common. Commercial projects often use outsourcing and subcontracting without careful and detailed oversight of the process.

A Software Development Process that Promoted High Quality

Based on their experiences and learning from past projects, a sophisticated software development process was created for the Shuttle. This development process was a major factor in the software success. Especially important was careful planning before any code was written, including detailed requirements specification; continuous learning and process improvement; a disciplined top-down structured development approach; extensive record keeping and documentation; extensive and realistic testing and code reviews; and detailed standards.

Extensive Planning and Specification. The Shuttle was one of the first spacecraft (and vehicles in general) to use a fly-by-wire flight control system, which created quality and reliability challenges. In response, NASA and its contractors developed a disciplined and structured development process. Increased emphasis was placed on the front end of development, including requirements definition, system design, standards definition, and top-down development.

An important feature of this process was extensive planning before starting to code: NASA controlled the requirements, and NASA and its contractors agreed in great detail on exactly what the code must do, how it should do it, and under what conditions before any code was produced. That commitment was recorded. Using those requirements documents, extremely detailed design documents were produced before a line of code was produced. Nothing was changed in the specifications (requirements or design) without agreement and understanding by everyone involved.

A common excuse used today for not writing requirements first is that the requirements are "unknown" or may change. In fact, in these cases, it is even *more* important to put major effort into upfront requirements analysis and specification. The software requirements for the Shuttle were continually evolving and changing, even after the system became operational and throughout its 30-year lifetime. NASA and its contractors made over 2,000 requirements changes between 1975 and the first flight in 1981. After the first flight, requirements changes continued. The number of changes proposed and implemented required a strict process to be used or chaos would have resulted.

The strategy used to meet the challenge of changing requirements had several components: rigorously maintained requirements documents, using a small group to create the software architecture and interfaces and then ensuring their ideas and theirs alone were implemented (called "maintaining conceptual integrity"), and establishing a requirements analysis group to provide a systems engineering interface between the requirements definition and software implementation worlds. The latter identified requirements and design trade-offs and communicated the implications of the trade-offs to both worlds. This strategy was effective in accommodating changing requirements without significant cost or schedule impacts.

The professional software development culture played a large role in the success of the Shuttle software.

All requested changes were submitted to the NASA Shuttle Avionics Software Control Board (SASCB). The SASCB ranked the changes based on program benefits including safety upgrades, performance enhancements, and cost savings. A subset of potential changes were approved for requirements development and placed on the candidate change list. Candidates on the list were evaluated to identify any major issues, risks, and impacts and then detailed size and effort estimates were created.

Once the change was approved and baselined, implementation was controlled through the configuration management system, which identified: the approval status of the change; the affected requirements functions; the code modules to be changed; and the builds (for example, operational increment and flight) for which the changed code was scheduled. Changes were made to the design documentation and the code as well as to other maintenance documentation used to aid traceability.

Detailed design specifications were developed only *after* the requirements specifications. Today in software development (and even touted as desirable by software researchers), design specifications are too often substituted for true requirements specifications or the two are mixed, making requirements analysis and management during development and over the life of the system extremely difficult.

When coding finally did begin, top-down development was the norm, using stubs and frequent builds to ensure interfaces were correctly defined and implemented first, rather than finding interface problems late in development during system testing. No programmer changed the code without changing the specification so the specifications and code always matched.

Due to the size, the complexity, the still-evolving nature of the requirements, and the need for software to help develop and test the Shuttle hardware, NASA and IBM created the software using incremental releases. Each release contained a basic set of capabilities and provided the structure for adding additional functions in later releases. These incremental releases were carefully planned to ensure later additions could be successfully integrated without requiring extensive changes. These planning and specification practices made maintaining software for over 30 years possible without introducing errors when changes were necessary.

Continuous Improvement. Continuous improvement was another critical feature of the software process. One of the guiding principles of the Shuttle software development was if a mistake was found, you should not just fix the mistake but must also fix whatever permitted the mistake in the first place. The process that followed the identification of any software error was: fix the error; identify the root cause of the fault; eliminate the process deficiency that let the fault be introduced and not detected earlier; and analyze the rest of the software for other, similar faults.

The goal was not to blame people for mistakes but instead to blame the process. The development process was a team effort; no one person was ever solely responsible for writing or inspecting the code. Thus there was accountability, but accountability was assigned to the group as a whole.

Carefully Defined Communication Channels. Such a large project and its long-term nature created communication problems. In response to the communication and coordination problems during Apollo development, NASA had created a control board structure, which was extended for the Shuttle. Membership on the review boards included representatives from all affected project areas, which enhanced communication among functional organizations and provided a mechanism to achieve strict configuration control. Changes to approved configuration baselines, which resulted from design changes, requirements change requests, and discrepancy reports, were coordinated through the appropriate boards and ultimately approved by NASA. Audits to verify consistency between approved baselines and reported baselines were performed weekly by the project office. In addition, the review checkpoints, occurring at critical times in development, that had been created for Apollo were again used and expanded.

Testing and Code Reviews. A final important feature of the development process with respect to achieving high quality involved extensive testing and code reviews: Emphasis was placed on early error detection, starting with requirements. Extensive developer and verifier code reviews in a moderated environment were used. It is now widely recognized that human code reviews are a highly effective way to detect errors in software, and they appear to have been very effective in this environment too.

A Professional Software Development Culture

Culture matters. There was a strong sense of camaraderie and a feeling that what they were doing was important. Many of the software developers worked on the project for a long time, sometimes their whole career. They knew the astronauts, many of whom were their personal friends and neighbors. These factors led to a culture that was quality focused and believed in zero defects.

The Shuttle software development job entailed regular 8 A.M. to 5 P.M. hours, where late nights were an exception. The atmosphere and the people were very professional and of the highest caliber. Words that have been used to describe them include businesslike, orderly, detail-oriented, and methodical. Smith and Cusumano note they produced "grownup software and the way they do it is by being grown-ups."[6]

The culture was intolerant of "ego-driven hotshots": "In the Shuttle's culture, there are no superstar programmers. The whole approach to developing software is intentionally designed not to rely on any particular person."[6] The cowboy culture that flourishes in some software development companies today was discouraged.

The culture was also intolerant of creativity with respect to individual

Calendar of Events

June 16–22
ACM SIGPLAN Conference on Programming Language Design and Implementation,
Seattle, WA,
Sponsored: SIGPLAN,
Contact: Hans-J. Boehm,
Email: hans.boehm@hp.com,
Phone: 650-857-3406

June 16–20
ACM Conference on Electronic Commerce,
Philadelphia, PA,
Sponsored: SIGecom,
Contact: Michael Kearns,
Email: mkearns@cis.upenn.edu,
Phone: 215-898-7888

June 16–21
The 22nd International Symposium on High-Performance Parallel and Distributed Computing,
New York, NY,
Sponsored: SIGARCH,
Contact: Manish Parashar,
Email: parashar@rutgers.edu,
Phone: 732-445-5388

June 17–19
ACM Information Hiding and Multimedia Security Workshop,
Montpellier, France,
Sponsored: SIGMM,
Contact: William Puech,
Email: william.puech@lirmm.fr

June 17–20
Creativity and Cognition 2013,
Sydney, NSW Australia,
Sponsored: SIGCHI,
Contact: Yukari Nagai,
Email: ynagai@jaist.ac.jp

June 17–20
Symposium on Computational Geometry 2013,
Rio de Janiero, Brazil,
Sponsored: SIGGRAPH, SIGACT,
Contact: Thomas Lewiner,
Email: lewiner@gmail.com

June 17–21
ACM SIGMETRICS/ International Conference on Measurement and Modeling of Computer Systems,
Pittsburgh, PA,
Sponsored: SIGMETRICS,
Contact: Mor Harchol-Balter,
Email: harchol@cs.cmu.edu

coding styles. People were encouraged to channel their creativity into improving the process, not violating strict coding standards. In the few occasions when the standards were violated, resulting in an error in the flight software, they relearned the fallacy of waiving standards for small short-term savings in implementation time, code space, or computer performance.

A larger number of women were involved in the Shuttle software engineering than is common in the software development world today. Many of these women were senior managers or senior technical staff. Smith and Cusumano[6] suggest the stability and professionalism may have been particularly appealing to women.

The challenging work, cooperative environment, and enjoyable working conditions encouraged people to stay with the Shuttle software project. As those experts passed on their knowledge, they established a culture of quality and cooperation that persisted throughout the program and the decades of Shuttle operations and software maintenance activities.

Limitations of the Process

No development process is perfect and the Shuttle software is no exception. Various external reviews identified gaps in the process that needed to be filled. One was that the verification and validation inspections by developers did not pay enough attention to off-nominal cases.

A second deficiency identified was a lack of system safety focus by the software developers and limited interactions with system safety engineering. System-level hazards were not traced to the software requirements, components, or functions.

A final identified weakness related to system engineering. The NRC committee studying Shuttle safety after the Challenger accident recommended that NASA implement better top-down, system engineering analysis, including system safety analysis.[7]

Conclusion

Software engineering seems to be moving in the opposite direction from the process used for the Shuttle software development.

The Shuttle software was not perfect, although it was better than most software today. Errors occurred in flight or were found in software that had flown. None of these software errors led to the loss of the Shuttle, although some almost led to the loss of expensive hardware and some did lead to not fully achieving mission objectives, at least by using the software: Because the orbital functions of the Shuttle software were not fully autonomous, astronauts or Mission Control could usually step in and manually recover from the few software problems that did occur. This too is a major lesson that should be learned by those rushing to make totally autonomous systems today.

The few errors in the flight software should not detract from the excellent processes used for the Shuttle software development. When errors were found, they were usually traced to temporary lapses in the rigorous processes or to periods of lowered morale. One takeaway is that there is more to achieving high quality than simply rigorous processes (as promoted by Taylorists in the guise of such process-heavy concepts as CMM and CMMI). The culture of the development environment may be just as important or maybe more so.

Beyond the lessons learned that have been noted so far, some general conclusions can be drawn from the Shuttle experience to provide guidance for the future. One is that high-quality software is possible but requires a desire to do so and an investment of time and resources. Software quality is often given lip service in many industries, where frequently speed and cost are the major factors considered, quality simply needs to be "good enough," and frequent corrective updates are the norm.

Software engineering seems to be moving in the opposite direction from the process used for the Shuttle software development, with requirements and careful pre-planning relegated to a less important position than starting to code. Strangely, in many cases, a requirements specification is seen as something that is generated after the software design is complete or at least after coding has started. Why has it been so difficult for software engineering to adopt the disciplined practices of the other engineering fields? There are still many software development projects that depend on cowboy programmers and "heroism" and less than professional engineering environments.

Ironically, many of the factors that led to success in the Shuttle software were related to limitations of computer hardware in that era, including limitations in memory that prevented today's common "requirements creep" and uncontrolled growth in functionality as well as requiring careful functional decomposition of the system requirements in order to break it into small pieces that could be loaded only when needed. Without these physical limitations that impose discipline on the development process, we need to determine how to impose discipline on ourselves and today's safety-critical projects. C

References

1. Broache, A. IRS trudges on with aging computers. CNET News (Apr. 12, 2007); http://news.cnet.com/2100-1028_3-6175657.html.
2. Eggan, D. and Witte, G. The FBI's upgrade that wasn't. *The Washington Post* (Aug. 18, 2006); http://www.washingtonpost.com/wp-dyn/content/article/2006/08/17/AR2006081701485.html.
3. Johnson, K. Denver airport saw the future. It didn't work. *New York Times*, (Aug. 27, 2005); http://www.nytimes.com/2005/08/27/national/27denver.html?pagewanted=all&_r=0.
4. Lewyn, M. Flying in place: The FAA's air control fiasco. *Business Week* (Apr. 26, 1993), 87, 90.
5. *Post-Challenger Evaluation of Space Shuttle Risk Assessment and Management.* Aeronautics and Space Engineering Board, National Research Council, January 1988.
6. Smith, S.A. and Cusumano, M.A. *Beyond the Software Factory: A Comparison of Classic and PC Software Developers.* Massachusetts Institute of Technology, Sloan School WP#3607=93\BPS, (Sept. 1, 1993).
7. Tomayko, J. *Computers in Spaceflight: The NASA Experience.* NASA Contractor Report CR-182505, 1988.

Nancy G. Leveson (leveson@mit.edu) is Professor of Aeronautics and Astronautics and also Professor of Engineering Systems at Massachusetts Institute of Technology, Cambridge, MA.

This column has been derived from the chapter "Software and the Challenge of Flight Control" of the book *Space Shuttle Legacy: How We Did It/What We Learned*, R. Launius, J. Craig, and J. Krige, Eds., 2013. The complete chapter can be downloaded from http://sunnyday.mit.edu/STAMP-publications.html.

DOI:10.1145/2461256.2461271

Stephen B. Wicker and Stephanie M. Santoso

Viewpoint
Access to the Internet Is a Human Right

Connecting Internet access with freedom of expression and creativity.

In a *New York Times* editorial,[2] Google Chief Internet Evangelist and ACM President Vinton G. Cerf asserted Internet access is not a human right. He argued that a given technology can enable a human right, but cannot itself be a human right. He went on to say it is a mistake to assert rights status for any technology, as technology is a means to an end, not the end or desired outcome itself. He used horses as an example; horses were once necessary to make a living. This does not, however, make access to horses a human right; rather, it is the ability to make a living that is the right. If we assert rights status for the horse specifically, or other enabling technologies more generally, we will end up valuing the wrong things.

We disagree. Human rights are a bundle that includes both an abstract expression of the right and some means for enabling that right. Freedom of the press, for example, is enshrined in the First Amendment to the U.S. Constitution. It would make little sense to say that press freedom is a human right, but that governments remain free to limit access to printing technology. Furthermore, something may be a human right, and yet still change over time. The technology available to Benjamin Franklin was dramatically different from that available to today's blogger, but that in no way mitigates Franklin's and the blogger's rights to their individual technologies under the general umbrella of freedom of the press. But the question remains: Is access to the Internet a human right? With which conceptual rights is it bundled?

The human rights logo (the dove-like hand on the left) was chosen via a crowdsourced online competition involving people from 190 countries submitting over 15,000 entries.

Rights, Freedom of Expression, and the Internet

The first step, of course, is to define the term "human right." The former Oxford Professor of Moral Philosophy James Griffin defines human rights as those aspects of our lives that are critical to our capacity to choose and to pursue our conception of a worthwhile life.[6] These aspects can be expressed in terms of a certain set of capabilities granted to the individual by society as a matter of justice. Amyarta Sen[16] and Martha Nussbaum[14] have pursued this idea in terms of what they call "the capabilities approach" to social justice. Nussbaum, for example, asserts that rights emerge from a consideration of the humanity of the individual. She starts with a "conception of the dignity of the human being and of a life that is worthy of that dignity, a life that has available in it 'truly human functioning'." She then proceeds to justify a list of 10 capabilities as central requirements of a life with dignity. For our purposes we focus on the first portion of the tenth capability, control over one's political environment:

> *Being able to participate effectively in political choices that govern one's life; having the right of political participation, protections of free speech and association.*
>
> —Martha C. Nussbaum, *Frontiers of Justice*[14]

Our argument for rights status for Internet access is that it is inextricably intertwined with the basic capability to participate effectively in political choices and to practice free speech and association. You cannot constrain access without damaging the rest.

We begin by connecting Internet access with those human goods that underlie rights status for freedom of expression. Arguments for the latter have deep roots; we go back almost 400 years to the poet John Milton and his response to the Licensing Order of 1643. In the *Areopagitica*,[12] Milton argued passionately (and floridly) against limitations on the freedom of the press, while making one of the first cogent arguments for freedom of expression to be found in English literature.[8] He claimed that free speech was valuable as a means for finding the truth, "a perfect shape most glorious to look on."

Two hundred years later, John Stuart Mill adopted Milton's argument and took it a step further. In *On Liberty*[11] he asserted that we cannot be certain that suppressed speech does not in fact contain the truth. To suppress speech is to assume we have complete knowledge of the truth, and do not need to hear what is being suppressed. Mill then added a dynamic element to the argument, asserting that even though the speech we suppress may be generally false, it may yet contain some kernel of truth. Our own position as to the case may not be entirely true, and may thus benefit from comparison and debate with other opinions that may contain their own partial truth. The resulting synthesis, emerging through debate fueled by free expression, will be more complete.

One of the leading scholars of the First Amendment in the middle of the past century, Thomas Emerson, completed the argument by connecting freedom of expression to personal well-being. In *Toward a General Theory of the First Amendment*,[5] Emerson included Milton and Mill's focus on finding the truth, but added self-development and societal participation as important arguments for freedom of expression. Drawing on a host of references that range from Milton, Locke, and Mill to the Frankfurt School psychoanalyst Erich Fromm, Emerson derived four "broad categories" of values that underlie protection of free expression:

> *The values sought by society in protecting the right to freedom of expression may be grouped into four broad categories. Maintenance of a system of free expression is necessary (1) as a means of assuring individual self-development, (2) as a means of attaining the truth, (3) as a method of securing participation by the members of society in social, including political, decision making, and (4) as a means of maintaining the balance between stability and change in the society.*
>
> —Thomas I. Emerson, *Toward a General Theory of the First Amendment*[5]

These four categories are clearly connected to Internet access. (1) The Internet offers a wide variety of means for self-development through experimentation, discovery, and the testing of one's opinions and beliefs, whether through social media, blogging, or commenting on articles in the digital editions of one's favorite newspapers. (2) The Internet enables the search for truth by providing access to an unparalleled amount of information.

The Internet advances freedom of expression in a manner and to an extent that dwarfs all other modes of communication.

From Wikipedia to the world's finest libraries to a wide array of document archives, there is an immense amount of material at one's fingertips when one has access to the Internet. (3) The Internet is a marvelous means for securing participation. In this sense, the Internet has redefined the public sphere. Jürgen Habermas defines the public sphere as "a network for communicating information and points of view."[7] Prior to the Internet, the hub-and-spoke architecture of mass media dictated the public sphere.[1] With the advent of the Internet, information no longer flows in only one or two directions, but full circle and in multiple directions, promoting discussion, dialogue, and debate. There has never been a grander ongoing conversation than the aggregate discussion that takes place every day on the Internet, a "conversation" in many media that covers every form of artistic expression imaginable for a wide variety of purposes. (4) The balance to which Emerson alludes is attained by providing mechanisms for individuals to vent their frustrations and reactions to change in open fora. The Internet certainly provides ample opportunity for such expression.

Beyond Enablement

The Internet is clearly a means for advancing the values that buttress the rights status of freedom of speech, but does that make Internet access a right in itself or just an enabler of rights? The final piece to our argument rests with the uniqueness of the Internet—it advances freedom of expression in a manner and to an extent that dwarfs all other modes of communication. To make our point, we consider the mode of public discourse that was supplanted by the Internet. Following Jean d'Arcy,[4] we classify broadcasting, advertising, and related attempts by large corporations to reach individuals as "vertical communication," while that from individuals to other individuals is termed "horizontal communication." Television, for example, is primarily a vertical communication medium and generally not a means for expression by individuals, while voice telephony or email is primarily horizontal.

The ability of vertical communication to shape public opinion has been noted for some time. In his 1922 book, *Public Opinion*,[10] American writer Walter Lippman asserted that the mass media plays a significant role in its formation. It is through the lens of the media, specifically the news and other types of information the media widely distribute, that members of society establish their views on cogent issues. In having such control, the mass media has the power to manufacture consent and develop propaganda.

The uniqueness of the Internet can also be argued from the negative: lack of Internet access has been shown to create a "digital divide."

Lippman notes that in order for propaganda to be created, "there must be a barrier between the public and the event." By enabling horizontal communication to an unparalleled extent, the Internet removes many of these barriers, facilitating the development of more objective public opinions, freer public discourse, and a less pliable electorate. Independent "bloggers," for example, have debunked prominent news stories while uncovering news that would not otherwise have been reported.[15] One need only note corporate media references to "bloggers in pajamas"[a] to get a sense of the former's frustration at its loss of dominance.

We should not stop, however, with the shaping of public opinion. The Internet is also a potent mechanism for developing what the Austrian philosopher Ivan Illich called a "convivial" lifestyle. Convivial living is an existence in society in which each individual has the ability to live with a certain amount of personal freedom and creativity. In order to have a convivial life, Illich asserted the need for people to have at their disposal convivial tools. These tools enable individuals to live autonomously by allowing them to leverage their individual potential, make their own decisions, express themselves, and generally exercise individual freedoms. The need for convivial tools emerges as a result of Illich's observation that as society becomes more industrialized, these tools, which can range from actual machinery or hand tools, to technology, to the skills and education required to operate machinery, become controlled by corporate elites and other powerful institutions who employ individuals to use these tools for the benefit of the bottom line:

> *In an age of scientific technology, the convivial structure of tools is a necessity for survival in full justice which is both distributive and participatory.... their central control in the hands of a Leviathan would sacrifice equal control over inputs to the semblance of an equal distribution of outputs. Rationally designed convivial tools have become the basis for participatory justice.*
>
> —Ivan Illich, *Tools for Conviviality*[9]

As a "convivial tool," the Internet has the potential to ensure the interests of the individual are preserved by enabling these interests to be publicly communicated, discussed, and debated. The Internet gives a voice to individuals, including marginalized populations, who might not otherwise have the ability to express their thoughts and opinions.

The uniqueness of the Internet can also be argued from the negative; lack of Internet access has been shown to create a "digital divide," a knowledge gap that leaves those without access to the Internet substantially less able to evaluate the candidates and propositions on offer in democratic institutions.[13] The digital divide has impact well beyond the political, as it creates disparities in social networking, in health care through telemedicine, and even in the individual's ability to make intelligent decisions about products and services.

In summary, access to the Internet is directly tied to a set of human capabilities that are considered fundamental to a life worth living. Access and these capabilities are so intertwined

[a] http://www.foxnews.com/story/0,2933,132494,00.html.

that one cannot deny rights status to Internet access without diminishing or denying the associated capabilities.

Consequences of Rights Status

If one accepts that Internet access is a human right,[b] then certain consequences must be acknowledged. But first we must avoid the hyperbole that may ensue from a rights discourse; there will be no suggestion here that the U.S. government should hand out computers while requiring that ISPs provide their services for free. Shelter, food, and the machinery of a free press are marketplace goods, so there is no reason to treat Internet access any differently. But as with those other fundamental goods, the government should implement a regulatory policy that recognizes Internet access as a human right.

A careful description of the right to Internet access is a first step. We treat it as having two parts: user access to an ISP, and the performance of all ISPs in carrying Internet content.

If Internet access is understood to be a human right, access to a wide variety of ISPs should be provided at nondiscriminatory rates. The current cable/DSL duopoly should be eliminated by requiring cable and DSL providers to allow access to other ISPs.[c] One should be able, for example, to access Brand X ISP services through one's Comcast cable modem, if one so chooses. This would create sufficient competition to ensure fair prices and quality of service. We note that the U.S. recently ranked 25th in the world in average Internet connection speeds[3]; a result, in part, of the current lack of competition. There should also be government aid for those unable to afford access, whether through computer services in schools, libraries, or similar mechanisms. Federal initiatives such as the Broadband Technology Opportunities Program (BTOP) and the Broadband Initiatives Program (BIP) are excellent steps in this direction.

Access to the Internet is directly tied to a set of human capabilities that are considered fundamental to a life worth living.

As for the performance of ISPs, a general common carrier rule should be in place: ISPs should not be allowed to block or discriminate with regard to price or quality of service based on the content being carried. As with most general rules, there may be exceptions, but the question arises as to who gets to decide whether the given content constitutes an exception. The First Amendment to the U.S. Constitution is very clear that Congress "shall make no law...abridging the freedom of speech," and yet the U.S. Supreme Court has found that some types of speech may be abridged. According to current jurisprudence, speech may be prohibited if that speech constitutes advocacy that is "directed to inciting or producing imminent lawless action."[d] Congress or the states may also pass laws that prohibit speech that is itself a violation of human rights, such as child pornography.[e]

It remains the case, however, that the First Amendment to the U.S. Constitution has its greatest impact when protecting speech that is considered repulsive to most listeners, for it is with such cases that settled ideas are put to their greatest test. Given the importance of this process, it should remain the province of democratic institutions and their designated courts to decide when content may be blocked. It should not be the province of service providers—entities whose interests may not be fully congruent with the underlying philosophy of the First Amendment—to develop their own policies as to what speech is acceptable and what is not.

It follows that an ISP providing variable quality of service to sites based on their content is violating the rights of its customers. Attempts to create markets in differential services should be recognized for what they are—a dangerous distortion of the Internet's unsurpassed capacity for expression. If such distortion is permitted, the vertical will be emphasized over the horizontal, and the Internet will slowly devolve into just another source of corporate programming. The capabilities proffered by the Internet are too important. Social justice demands that our access to the Internet and its content not be left to the vagaries and potential abuses of the marketplace.

[b] In January 2013 Germany's Federal Court of Justice declared that Internet access was a basic human right. The Court's argument was based on the Internet's importance to everyday life, and the "significant impact" on the individual when it is absent. See http://www.dw.de/internet-access-declared-a-basic-right-in-germany/a-16553916.

[c] In particular, the FCC's categorization of broadband service providers as "information services" as defined by the Telecommunications Act of 1996—a decision that relieves these service providers from having to follow Title II common carrier requirements—should be reversed.

[d] *Brandenburg v. Ohio*, 395 U.S. 444 (1969).

[e] *United States v. Williams*, 553 U.S. 285 (2008).

References

1. Benkler, Y. *The Wealth of Networks: How Social Production Transforms Markets and Freedom.* Yale University Press, New Haven, CT, 2006.
2. Cerf, V.G. Internet access is not a human right. *New York Times* (Jan. 4, 2012); http://www.nytimes.com/2012/01/05/opinion/internet-access-is-not-a-human-right.html?_r=0.
3. Communication Workers of America. *A Report on Internet Speeds in All 50 States*, 2010; http://www.speedmatters.org.
4. D'Arcy, J. An ascending progression. In *The Right to Communicate: A New Human Right.* D. Fisher and L.S. Harms, Eds., Boole Press, Dublin, 1982.
5. Emerson, T.I. *Toward a General Theory of the First Amendment.* Random House, New York, 1963.
6. Griffin, J. *On Human Rights.* Oxford University Press, 2008.
7. Habermas, J. Between facts and norms. *Contributions to a Discourse Theory of Law and Democracy.* Polity Press, Cambridge, MA, 1996.
8. Haworth, A. *Free Speech.* Routledge, London, 1998.
9. Illich, I. *Tools for Conviviality.* Harper & Row, New York, 1973.
10. Lippman, W. *Public Opinion.* Macmillan, New York, 1922.
11. Mill, J.S. *On Liberty.* Longman, Roberts & Green, London, 1869.
12. Milton, J. *Areopagitica.* London, 1644.
13. Norris, P. *Digital Divide: Civic Engagement, Information Poverty, and the Internet Worldwide.* Cambridge University Press, 2001.
14. Nussbaum, M.C. *Frontiers of Justice: Disability, Nationality, and Species Membership.* Belknap, Harvard, 2006.
15. Rosenberg, S. *Say Everything: How Blogging Began, What It's Becoming, and Why It Matters.* Three Rivers Press, 2009.
16. Sen, A.K., Ed. *Commodities and Capabilities.* Oxford University Press, 1985.

Stephen B. Wicker (wicker@ece.cornell.edu) is a professor in the School of Electrical and Computer Engineering at Cornell University, Ithaca, NY.

Stephanie M. Santoso (sms629@cornell.edu) is a Ph.D. candidate in the Information Science department at Cornell University, Ithaca, NY.

This work was funded in part by the U.S. National Science Foundation TRUST Science and Technology Center and the NSF Trustworthy Computing Program. The authors gratefully acknowledge the comments from and arguments with Adam and Tonya Engst, Steven Jackson, Andreas Kuehn, David Lundie, Daniel Paré, Murali Venkatesh, and Sarah Wicker.

DOI:10.1145/2461256.2461273

Article development led by acmqueue
queue.acm.org

Risk is a necessary consequence of dependence.

BY DAN GEER

Resolved: The Internet Is No Place for Critical Infrastructure

WHAT IS CRITICAL? To what degree is critical defined as a matter of principle, and to what degree is it defined operationally? I am distinguishing what we say from what we do.

Mainstream media love to turn a spotlight on anything they can label "hypocrisy," the *Merriam-Webster Unabridged Dictionary* meaning of which is:

"[T]he act or practice of pretending to be what one is not or to have principles or beliefs that one does not have, especially the false assumption of an appearance of virtue."

The debate topic I propose here can therefore be restated as calling out, "Hypocrisy!" on the claim that the Internet is a critical infrastructure either directly or by transitive closure with the applications that run on or over it. If the claim were true, the divergence between our beliefs and our practices would be necessarily narrower (by *our* I mean each of us both separately and collectively).

Perhaps I am echoing how a free-range cattleman felt about the coming of barbed wire, roads, and land title to the American West. The great cattle drives of the West lasted 20 years before other kinds of progress made them impossible. Commercial Internet traffic began some 20 years ago, with the interconnection of PSInet and UUNet by Commercial Internet Exchange (CIX).

PHOTOGRAPH BY TIM BOOTH

Recalling Winston Churchill's "The further back I look, the further forward I can see," either the wide open range that is the freedom of an Internet built on the end-to-end principle must die, or else we must choose not to allow the critical infrastructure of our lives to depend on that Internet. Freedom and reliability are now at odds.

Consider the Internet as a Hobson's choice: either you get it, warts and all, or you get nothing. According to The Pew Research Center's Internet and American Life Project:[13]

"One in five American adults do not use the Internet. Among adults who do not use the Internet, almost half [said] that the main reason they don't go online is because they don't think the Internet is relevant to them."

For those 10% who, presented with a take-it-or-leave-it proposition regarding the Internet, choose "leave it," the Internet does not register as desirable and may, for some of them, be undesirable.

Opting Out Is Hardly an Option

I have never bought or owned a television. There is no social opprobrium if you opt out of television; it is merely a choice. That 10% of the population that does not bother with the Internet is surely similar to whatever fraction of the population does not see any reason to bother with a television. But can they refuse the Internet as a *choice*, one that is inconsequential to their lives, the way television is inconsequential to mine?

No. It is not possible to live your life without having a critical dependence on the Internet, even if you live at the end of a dirt road but still occasionally buy nails or gasoline. Unlike television, you cannot unplug from the Internet even if you want to. If you are dependent on those who are dependent on television, then so what? If, however, you are dependent on those who are dependent on the Internet, then so are you. Dependence with respect to television is not transitive. Dependence with respect to the Internet is transitive.

Those who choose to "leave it" are still dependent on it unless they are living a pre-industrial life. That rejectionists depend on people who are not rejectionist is simply a fact.

But rejectionists do have impact—they are now a kind of fail-safe. If we begin to penalize the rejectionists—that is, force them to give up on their rejectionism—we will give up a residuum of societal resiliency.

To illustrate, I have a 401(K) account with Fidelity Investments. Fidelity no longer accepts client instructions in writing; it only accepts instructions over the Internet or, as a fallback for the rejectionist, over the phone. It simply does not accept the canonical wet ink signature on bond paper. I have mailed Fidelity postal letters, and its representatives have responded in email messages that explain just that (though I should note that I never gave them my email address). Fidelity's stand is that its auditors approve of this scheme. My stand is, "Your auditors work for you, not me." Those email letters do not contain a digital signature and, in any case, what is the equivalent of that for a phone call? Fidelity still sends paper statements to the same mailing address from which I have been writing.

I use a small local bank. I sent it a letter stating that as I would not be using online services, I would like the bank to turn off access to my account and raise an alarm if anyone ever tried to use the uninitialized account waiting in my name. The bank agreed without any argument. That is not the norm. Try, as I have done, to make that same request to the arm of the payroll services giant ADP that runs the get-your-W2-online service called iPay. It will refuse.

Estonia is the most Internet-dependent country,[7] and Estonian pride is entirely in order. Its degree of dependence happens not to be for me: I want to retain the ability to opt out of direct dependence on the Internet—that is, to opt out of that dependence that is the root of risk. I mean that as stronger than a preference but weaker than an ultimatum.

In a free society, that which is not forbidden is permitted. In a non-free society, that which is not permitted is forbidden. Obamacare deploys the government's monopoly on the use of force to collectivize the downside risk of illness. Just as forcibly collectivizing the downside risk of illness has its utopian proponents, so, too, does forcibly collectivizing the downside risk of Internet exposure.

Estonia is well ahead of nearly everybody in intentional dependence on the Internet; China is well ahead of nearly everybody in forcibly collectivizing the extent and manner in which its citizens use the Internet. As sovereigns, the former is Estonia's right just as the latter is China's right. I want neither, even though I must acknowledge that as nations decide on their particular mix of dependencies, the Internet will be dramatically Balkanized. The Internet will never again be as free as it is today.

In 2002, a total computer outage occurred at Harvard's Beth Israel Hospital.[6] The event was severe, unexpected, and recovery was frustrated by complexity. That a fallback to manual systems was possible saved the day, and it was those who could comfortably work without network dependence who delivered on that possibility, because they were old enough to have done so previously.

Risk is a consequence of dependence. Because of shared dependence, aggregate societal dependence on the Internet is not estimable. If dependencies are not estimable, then they will be underestimated. If they are underestimated, then they will not be made secure over the long run, only over the short. As the risks become increasingly unlikely to appear, the interval between events will grow longer. As the latency between events grows, the assumption that safety has been achieved will also grow, thus fueling increased dependence in what is now a positive feedback loop. Accommodating rejectionists preserves alternative, less complex, more durable means and therefore bounds dependence. Bounding dependence is the core of rational risk management.

Common-Mode Failure

In the language of statistics, *common-mode failure* comes from underappreciated mutual dependence. Quoting the National Institute of Standards and Technology:[9]

"[R]edundancy is the provision of functional capabilities that would be unnecessary in a fault-free environment. Redundancy is necessary, but not sufficient for fault tolerance... System failures occur when faults propagate to the outer boundary of the system. The goal of fault tolerance is to intercept the propagation of faults so that failure does not occur, usually by substituting redundant functions for functions affected by a particular fault. Occasionally, a fault may affect enough redundant functions that it is not possible to reliably select a non-faulty result, and the system will sustain a common-mode failure. A common-mode failure results from a single fault (or fault set). Computer systems are vulnerable to common-mode resource failures if they rely on a single source of power, cooling, or I/O. A more insidious source of common-mode failures is a design fault that causes redundant copies of the same software process to fail under identical conditions."

That last part—"A more insidious source of common-mode failures is a design fault that causes redundant copies of the same software process to fail under identical conditions"—is exactly that which can be masked by complexity, precisely because complexity ensures underappreciated mutual dependence.

Which brings us to critical infrastructure and the interconnection between critical infrastructures by way of the Internet. Quoting the Clinton Administration's Policy on Critical Infrastructure Protection from May 22, 1998:

"Critical infrastructures are those physical and cyber-based systems essential to the minimum operations of the economy and government."[11]

"Essential to minimum operations" is not a requirement that the armor deflect all bullets, only that no bullet is paralyzing. One of the great Allied victories of World War II was getting 338,000 soldiers off the beaches of Dunkirk using 800 "little boats," a paragon of the phrase "essential to minimum operations."

The Internet is a network of networks, its main protocols designed for tolerance to random faults and for the absence of common-mode failure. It has been proven in practice.[2] It was not designed, however, for resistance to targeted faults, which cannot be done at the same time as you are designing for resistance to random faults.[1]

In an Internet crowded with important parts of daily life, the chance of common-mode failure is no idle worry. The Obama administration is broadly committed to increasing dependence on the Internet, most notably on two fronts: electronic health records and the so-called smart grid, either of which might be said to be "essential to the minimum operations of the economy and government." As with most garden paths, both can have eminently useful results for which a desire is rational. Both illustrate my point.

Electronic health records depend on the smooth functioning of electric power, networks, computers, displays, and a host of security features particularly as they relate to maintaining consistency across multiple practices.[12] The smart grid depends on almost everything we now know about power, including the absolute necessity of good clocks, a wide range of industrial controls operated flawlessly at distance and guaranteed not to lie about their state, and another host of security features. Both of these involve new levels of exposure to common-mode risk. Doing without their benefits will be easier for those who can remember not having had them.

Each new dependence raises the magnitude of downside risk, the potential for collateral damage, and the exposure of interrelationships never before realized. Forget the banks, it is the Internet that is too big to fail. While there is no entity that can bail out the Internet, there is no meaningful country that is not developing ways to disrupt the Internet use of its potential adversaries. The most a country might do is preserve the Internet interior to itself—as Estonia demonstrated when under attack from Russia—though at some level of trans-border interconnection, the very concept of "interior" loses meaning.

Designing for tolerable failure modes is precisely what security engineering is fundamentally about. The failure mode you did not think of will not be in your design; therefore, whether it is tolerable will depend on other things. The question, then, is: Can tolerable failure modes be designed? In other words, can a failure mode never before possible be added to the system such that larger, intolerable failures can be precluded? Is there a cyber-critical infrastructure analog to a shear-bolt in the drivetrain of heavy machinery?

No country, no government, no people need rules against things that are impossible. Our onrushing dependence on things never before possible creates vacua where, in the fullness of time, there will have to be rules.

No country, no government, no people need rules against things that are impossible. Our onrushing dependence on things never before possible creates vacua where, in the fullness of time, there will have to be rules. As Chicago Mayor Rahm Emanuel put it, the creation of rules is easier in a time of crisis, so one must "never let a good crisis go to waste." He is right as a matter of observation; he is wrong as a matter of probity. Just as driving under the influence of alcohol is wrong, so is making policy under the influence of adrenaline. Eleven years before he became the fourth president of the U.S., James Madison said:

"Perhaps it is a universal truth that the loss of liberty at home is to be charged to provisions against danger, real or pretended, from abroad."

One wonders how Madison would feel about an interconnected world where *abroad* has so thoroughly lost its meaning, at least with respect to Internet-dependent critical infrastructure if not national frontiers. My guess is that Madison would decide the Internet is, per se, "abroad." As such, our critical infrastructure is now another country, something from which to be protected at the loss of liberty.

I have previously spoken on whether having people in the loop for security is a fail-safe or a liability.[3] I will not recount the arguments here, but I will give my conclusion: a good security design takes people out of the loop except when it cannot and, when it cannot, it is clear that this is so. Putting a human in the loop—that is to say, falling back from automation—has proven to be a breakthrough finesse.

That the public has "volunteered" its unused computing power to botmasters is a historical mirror of how press gangs once filled the rosters of the British Navy, but how is that meaningfully different from a formal mandate that if you have medical records those shall be electronic, or if you receive electricity that the meter be a surveillance tool? How is it different from finding that compliance auditors have certified to distant regulators that there is no need to accept a signed paper letter detailing the wishes of the financial client?

What Price Security?

Security is a necessary but insufficient condition for reliability. As such, connecting the insecure (and thus unreliable) to the important and expecting the mélange to be reliable is utter foolishness. As network security expert Marcus Ranum says, "A system that can be caused to do undesigned things by outsiders is not 'reliable' in any sense of the word." Work being done by Sergey Bratus, Meredith Patterson, and others at Language-theoretic Security (LANGSEC) yields insight deserving full quotation:[8]

"The Language-theoretic approach regards the Internet insecurity epidemic as a consequence of ad hoc programming of input handling at all layers of network stacks, and in other kinds of software stacks. LANGSEC posits that the only path to trustworthy software that takes untrusted inputs is treating all valid or expected inputs as a formal language, and the respective input-handling routines as a recognizer for that language. The recognition must be feasible, and the recognizer must match the language in required computation power.

When input handling is done in [an] ad hoc way, the de facto recognizer, i.e., the input recognition and validation code ends up scattered throughout the program, does not match the programmers' assumptions about safety and validity of data, and thus provides ample opportunities for exploitation. Moreover, for complex input languages the problem of full recognition of valid or expected inputs may be UNDECIDABLE, in which case no amount of input-checking code or testing will suffice to secure the program. Many popular protocols and formats fell into this trap, the empirical fact with which security practitioners are all too familiar.

Viewed from the venerable perspective of Least Privilege, ... computational power is privilege, and should be given as sparingly as any other kind of privilege to reduce the attack surface. We call this ... the Minimal Computational Power Principle.

We note that recent developments in common protocols run contrary to these principles. In our opinion, this heralds a bumpy road ahead. In particular, HTML5 is Turing-complete, whereas HTML4 was not."

Security is a necessary but insufficient condition for reliability. As such, connecting the insecure (and thus unreliable) to the important and expecting the mélange to be reliable is utter foolishness.

Nearly nothing we have in our cyber interfaces to critical infrastructure meets LANGSEC's test. Attaching the cyber interface of critical infrastructure to the Internet is a flat-out guarantee of error. Such error may be improbable, but probabilistic events eventually occur. If we are especially unlucky, those errors will not be prompt.

There has been much talk about whether to grant the U.S. president a kill switch for the Internet. There is some logic to that if, due to interdependence that is inestimable, it is not possible to disambiguate friend from foe. Were someone on an inbound airplane found to have smallpox, the passengers and crew would be quarantined as a matter of public health until each of them could be separately certified as disease free. Many important enterprises, public and private, quarantine inbound email with nearly as much vigor as they quarantine inbound DHL packages. The logic is sound. The time scale is human.

We have amongst ourselves, and we accommodate, cloistered communities such as the Amish. If a food crisis were to materialize, it is the Amish who would be least affected. We also have amongst ourselves neo-Luddites, who know where machines will lead and on that basis may well mimic their progenitors. The Amish merely wish to be left alone. Is there not room in our increasingly wired world for those who choose merely to be left alone, in this case choose not to participate in the Internet society? Do those who do not participate deserve to not have their transactions of all sorts be exposed to a critical infrastructure dependent on the reliability of Internet applications?

The U.S.'s ability to project power depends on information technology, and, as such, cyber insecurity is the paramount national security risk.[4] Putting aside an Internet kill switch, might it be wise for the national authorities to forbid, say, Internet service providers from propagating telnet, SSH v1, or other protocols known to be insecurable? If not, should cyber components of the Defense Industrial Base be forbidden to accept such connections? There is a freedom-vs.-reliability collision in that—if not a natural policy. There is a direct historical echo as well; in 1932 the foremost political commentator of the age, Walter Lippmann, told President Roosevelt, "The situation is critical, Franklin. You have no alternative but to assume dictatorial powers."

Again, when 10% of the population sees nothing in the Internet for them, should we respect that wish and ensure that, as with the Amish, there is a way for them to opt out without having to live in a cave? Should we preserve manual means for them?

I say yes, and I say so because the preservation of manual means is a guarantee of a fallback that does not have a common-mode failure with the rest of the interconnected, mutually vulnerable Internet world.

My colleague and I run the Index of Cyber Security.[5] Our respondents all have direct operational responsibility for cyber security. The Index is rising—that is, experts say risk is accumulating in much the same way that burnable timber accumulates on the eastern slope of the Rockies. This is a formal, metrics-based backstop to saying that "we" are not running fast enough to stay in the same place; therefore, preserving fallback is essential.

Department of Defense thinkers agree; their goal is no longer intrusion prevention but intrusion tolerance. If we are to practice evidence-based medicine on the body Internet, we first acknowledge that expensive therapy is not always the answer. Cost-effective medicine cannot be practiced if every human life is infinitely valuable. Perhaps you can come up with a cyber analogue to "quality-adjusted life years" and help us all decide when to treat, when to palliate, and when to accept mortality.

The following ideas from the Homeland Security Watch blog may be ones whose time has come:[10]

"The deficient and unforgiving design that many of us—private citizens, as well as public safety agencies—have adopted is dependence on just-in-time information.

My twenty-something children seldom preplan in any significant way. They expect cellphones, text messaging, Facebook, and email to allow them to seize the best opportunities that unfold. It works and I envy them. Except when it does not work. Except when these digital networks fail.

Much of our consumer culture is built around the same approach. We have become an economy, a society optimized for just-in-time. It can be a beautiful dance of wonderful possibilities emerging in a moment and rapidly synchronized across time and space. Until the music stops.

...There is a shared overconfidence in the fail-safe capabilities of protective design and effective communications. [T]he design bias increase[s] risk exposure, communications was confusing or worse, and both the design and the communications protocols complicate effective human response once risk [is] experienced."

Summing Up

Risk is a consequence of dependence. Because of shared dependence, aggregate societal dependence on the Internet is not estimable. If dependencies are not estimable, then they will be underestimated. If they are underestimated, then they will not be made secure over the long run, only over the short. As the risks become increasingly unlikely to appear, the interval between events will grow longer. As the latency between events grows, the assumption that safety has been achieved will also grow, fueling increased dependence in what is now a positive feedback loop. If the critical infrastructures are those physical and cyber-based systems essential to the minimum operations of the economy and government, and if leading cybersecurity operational management says risk is growing steadily, then do we divert more of our collective power to forcing security improvements that will be sharply diseconomic, or do we preserve fallbacks of various sorts in anticipation of events that seem more likely to happen as time passes?

Does "use it up, wear it out, make it do, or do without" have any meaning for us? Is centralizing authority the answer, or is avoiding further dependence the better strategy? Can we imagine starting over in any real sense, or is Balkanization not just for nations but for critical sectors as well? Is the creative destruction that is free enterprise now to be focused on remaking what are normally the steadying flywheels of American society, by which I mean government and other capital-intensive industries? Do we celebrate the individual who still prefers to fix things he or she already has, or are those individuals to be herded into national health information networks, smart grids, and cars that drive themselves?

Related articles on queue.acm.org

The Code Delusion
Stan Kelly-Bootle
http://queue.acm.org/detail.cfm?id=1317411

National Internet Defense—Small States on the Skirmish Line
Ross Stapleton-Gray, Bill Woodcock
http://queue.acm.org/detail.cfm?id=1929325

Communications Surveillance: Privacy and Security at Risk
Whitfield Diffie, Susan Landau
http://queue.acm.org/detail.cfm?id=1613130

References

1. Barabasi, L. and Albert, R. Emergence of scaling in random networks. *Science 286* (Oct. 1999), 509–512.
2. Branigan, S. and Cheswick, B. The effects of war on the Yugoslavian Network, 1999; http://www.cheswick.com/ches/map/yu/index.html.
3. Geer, D. People in the loop: Are they a fail-safe or a liability? Suits & Spooks (Feb. 8, 2012); http://geer.tinho.net/geer.suitsandspooks.8ii12.txt.
4. Hathaway, M. Securing our digital future. White House blog; http://www.whitehouse.gov/blog/2009/05/29/securing-our-digital-future.
5. Index of Cyber Security; http://cybersecurityindex.org.
6. Kilbridge, P. Computer crash—lessons from a system failure. *New England Journal of Medicine 348*, 10 (2003), 881–882; http://ehealthecon.hsinetwork.com/NEJM_downtime_2003-03-06.pdf.
7. Kingsley, P. How tiny Estonia stepped out of USSR's shadow to become an Internet titan. *The Guardian*; http://www.guardian.co.uk/technology/2012/apr/15/estonia-ussr-shadow-internet-titan.
8. LANGSEC: Language-theoretic Security; http://langsec.org.
9. National Institute of Standards and Technology. High Integrity Software System Assurance. Redundancy management (Sec. 4.2). A Conceptual Framework for System Fault Tolerance, 1995; http://hissa.nist.gov/chissa/SEI_Framework/framework_16.html.
10. Palin, P. Can you envision a "successful failure?" Homeland Security Watch; http://www.hlswatch.com/2012/07/13/can-you-envision-a-successful-failure/.
11. Presidential Decision Directive 63. The Clinton Administration's Policy on Critical Infrastructure Protection (1998); http://www.fas.org/irp/offdocs/paper598.htm.
12. Shim, S.S.Y. The CAP theorem's growing impact. *IEEE Computer 45*, 2 (2012), 21–22.
13. Zickuhr, K. and Smith, A. Digital differences. Pew Research Center; http://pewinternet.org/~/media//Files/Reports/2012/PIP_Digital_differences_041312.pdf.

Dan Geer is a security researcher with a quantitative bent. His group at MIT produced Kerberos, and after time at a number of startups, he is still at it—today as CISO at In-Q-Tel. A prolific writer, he authored the controversial paper on whether a computing monoculture rises to the level of a national security risk. Geer is an electrical engineer, a statistician, a farmer, and someone who thinks truth is best achieved by adversarial procedures.

DOI:10.1145/2461256.2461272

Article development led by acmqueue
queue.acm.org

Real-time finite difference-based sound synthesis using graphics processors.

BY BILL HSU AND MARC SOSNICK-PÉREZ

Real-Time GPU Audio

TODAY'S CPUS ARE capable of supporting real-time audio for many popular applications, but some compute-intensive audio applications require hardware acceleration. This article looks at some real-time sound-synthesis applications and shares the authors' experiences implementing them on graphics processing units (GPUs).

Software synthesizers, which use software to generate audio in real time, have been around for decades. They allow the use of computers as virtual instruments, to supplement or replace acoustic instruments in performance. Groups of software instruments can be aggregated into virtual orchestras. Similarly, in a video game or virtual environment with multiple sound sources, software synthesizers can generate each of the sounds in an auditory scene. For example, in a racing game, each discrete software synthesizer may generate the sound of a single car, with the resulting sounds combined to construct the auditory scene of the race.

Traditionally, because of limited computing power, approaches to real-time audio synthesis have focused on techniques to compute simple waveforms directly (for example, additive, FM synthesis), using sampling and playback (for example, wavetable synthesis) or applying spectral modeling techniques (for example, modal synthesis) to generate audio waveforms. While these techniques are widely used and understood, they work primarily with a model of the abstract sound produced by an instrument or object, not a model of the instrument or object itself. A more recent approach is physical modeling-based audio synthesis, where the audio waveforms are generated using detailed numerical simulation of physical objects or instruments.

In physical modeling, a detailed numeric model of the behavior of an instrument or sound-producing object is built in software and then virtually "played" as it would be in the real world: the "performer" applies an excitation to the modeled object, analogous, for example, to a drumstick striking a drumhead. This triggers the computer to compute detailed simulation steps and generate the vibration waveforms that represent the output sound. By simulating the physical object and parameterizing the physical properties of how it produces sound, the same model can capture the realistic sonic variations that result from changes in the object's geometry, construction materials, and modes of excitation.

Suppose you are simulating a metallic plate to generate gong or cymbal-like sounds. Varying a parameter that corresponds to the stiffness of the material may allow you to produce sounds ranging from a thin, flexible plate to a thicker, stiffer one. By changing the surface area for the same object, you can generate sound corresponding to cymbals or gongs of different sizes. Us-

This image, by Syed Reza Ali, was created by an interactive real-time audio reactive visual application.

IMAGE BY SYED REZA ALI; WWW.SYEDREZAALI.COM

ing the same model, you may also vary the way in which you excite the metallic plate—to generate sounds that result from hitting the plate with a soft mallet, a hard drumstick, or from bowing. By changing these parameters, you may even simulate nonexistent materials or physically impossible geometries or excitation methods.

There are various approaches to physical modeling of sound synthesis. One such approach, studied extensively by Stefan Bilbao,[1] uses the *finite difference approximation* to simulate the vibrations of plates and membranes. The finite difference simulation produces realistic and dynamic sounds (examples can be found at http://unixlab.sfsu.edu/~whsu/FDGPU). Real-time finite difference-based simulations of large models are typically too computationally intensive to run on CPUs. In our work, we have implemented finite difference simulations in real time on GPUs.

Next, we address key issues in real-time audio software and look at how they relate to the computing characteristics of GPUs. We will look at how some types of real-time synthesis applications have benefited from GPU acceleration, ending with the challenges encountered while running finite difference-based synthesis on GPUs.

Real-Time Audio Generation in Software

Real-time audio synthesis can be broadly broken down into three steps:

1. *Excitation.* An excitation event signals the synthesizer that real-time audio generation should begin. To strike a virtual cymbal, for example, a human performer may hit a key on a keyboard, which generates an excitation event.

2. *Sample generation.* Audio sample data is computed for the desired sounds (for example, the cymbal crash).

3. *Output.* Data generated in step 2 is sent to system software for playback by the system.

Figure 1 shows two approaches to real-time audio synthesis: in 1a the naïve approach computes and outputs a single sample at a time, while in 1b the buffered approach computes multiple samples and outputs them as a block.

In Figure 1a, after the excitation

event, a function is called to generate a single sample. The new sample is sent to a buffer in the audio-output device. These two steps are repeated until a new excitation is received, or until the sound becomes inaudible. At the CD-quality sample rate of 44.1kHz, one sample has to be computed and ready for output every 1/44,100 seconds, or every 23µs.

The naïve approach in Figure 1a incurs high overhead. Every sample requires a function call and a copy into a system buffer, which may involve a context switch.

Instead, the buffered approach illustrated in Figure 1b is usually used. Samples are generated and transferred in blocks of *n* samples, significantly reducing overhead. Though the buffered approach reduces overhead, it introduces *latency* into the signal path. Latency is the time it takes from the excitation of the instrument to the production of its sound. The longer the latency, the less responsive an instrument feels. For software instruments, latency should be kept to a minimum, on the order of tens of milliseconds.

For the buffered approach, a block of *n* samples has to be generated in *n* × 23µs. In commonly used, less compute-intensive algorithms such as wavetable synthesis, generating a sample involves a few arithmetic operations and table lookups; this is usually achievable on the CPU. For compute-intensive real-time audio applications, the sample generation needs to be parallelized. The ubiquity of GPUs today makes them an obvious choice for such applications.

Figure 1. Real-time audio-synthesis approaches.

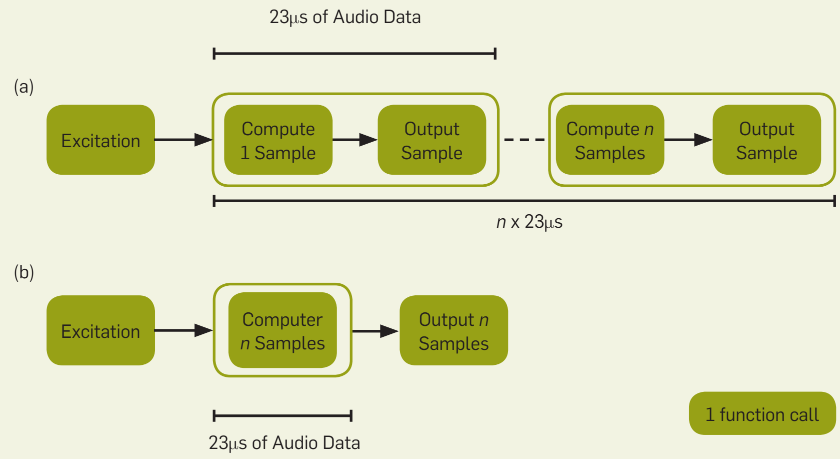
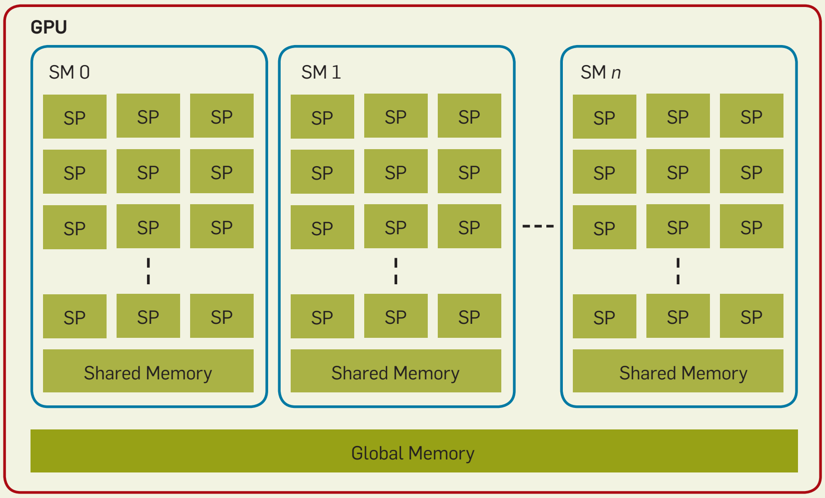


Figure 2. Hardware configuration of an NVIDIA GPU.



GPU Basics

NVIDIA's GPUs and CUDA platform are popular options for high-performance computing today. An NVIDIA GPU (Figure 2) is a hierarchical multiprocessor on a chip, usually consisting of a number of *streaming multiprocessors* (SMs); each SM contains a number of *streaming processors* (SPs). An SM can execute large numbers of threads simultaneously, with each thread running the same program. This *single instruction multiple thread* (SIMT) architecture is especially suitable for applications with a high degree of data parallelism, where the same operations are applied to large amounts of data. For example, the NVIDIA GeForce GT 650M in a mid-2012 MacBook Pro Retina has two SMs, each with 192 SPs (cores), at a clock rate of 900MHz.

To the CPU, an NVIDIA GPU looks like a separate coprocessor with its own memory system. Jobs are configured on the CPU, or *host*, which then interacts with the GPU *device*. The host copies data and device-specific programs from host memory to device memory, and initiates program execution on the device. The GPU then executes the jobs independently of the host, either synchronously or asynchronously. When the GPU device is done, results are copied from the device to host memory. NVIDIA CUDA systems provide ways of reducing this memory copy latency using such techniques as shared memory pages between the host and device.

A function that is executed in parallel by the GPU is called a *kernel*. Just as the GPU hardware is composed of streaming processors grouped into streaming multiprocessors, a kernel is executed in parallel by *threads* grouped into *blocks* (see Figure 3). One or more blocks are assigned to a streaming multiprocessor, guaranteeing that threads in the same block are executed on the same SM. Each thread executes the same kernel function, but usually on different data. Since threads in a block execute on the same SM, they can leverage fast hardware-supported synchronization and share data using shared memory within the same SM. Different blocks cannot be synchronized within a kernel and are not guaranteed execution order by any particular SM.

A hierarchy of memory is available on the GPU device. In addition to registers accessible to a thread, a limited amount of faster *shared memory* can be shared among threads in a block, but persists only as long as the SM is executing the thread block. Larger amounts of slower *global memory* can be accessed and shared among all threads in any block. Global memory is allocated on the device by the host and persists until deallocated by the host, but access times for global memory are slower than for shared memory. Optimized GPU code leverages these thread and memory characteristics.

In older GPUs, efficient kernel execution usually requires careful management of shared memory in software. More recent GPUs, based on the Fermi and Kepler architectures, support a true hardware cache architecture; explicit software management of shared memory is not as critical on these systems.

GPU-based Applications with Multiple Independent Audio Streams

GPUs have previously been used successfully in real-time audio applications. Many of these applications have involved the simultaneous generation of multiple loosely coupled sound sources or processing streams. In these instances the GPU has been used to ease the load on the CPU, caused by the computational complexity of generating and processing many sounds simultaneously. Examples include rendering and spatializing sound-generating objects in a game or virtual environment, or synthesizing multiple instruments in a virtual ensemble. Each sound source might be a car in a racing game or an instrument in an orchestra.

Recall the buffered approach in Figure 1b; at each computation step, for each sound source, a block of *n* samples is computed and sent to system buffers for playback. Sequential samples in a buffer usually have to be computed in order. Since buffers for two sound sources can be computed independently of each other, assigning each sound source to a thread in the GPU is straightforward. Each thread computes an *n*-sample buffer and synchronizes; output from all the sources are mixed down and sent to system buffers for playback (see Figure 4). Since a typical GPU today efficiently supports thousands of simultaneous threads, this type of application is a good match for GPU acceleration.

For example, Zhang et al.[3] describe a typical parallel setup that implements *modal synthesis*; the sound of a vibrating object is generated by combining a bank of damped sinusoidal oscillators, each representing a *mode*, or resonant frequency of the sound. Since the *n* samples for each mode can be calculated independently of other modes, each mode can be assigned to an independent thread in the GPU. After *n* samples are generated for each mode, the results for all modes are combined for playback.

Real-time finite difference synthesis works somewhat differently and is arguably not an efficient use of the GPU, but we have been able to get useful results despite some severe constraints.

Figure 3. Block and thread configuration of an NVIDIA GPU.

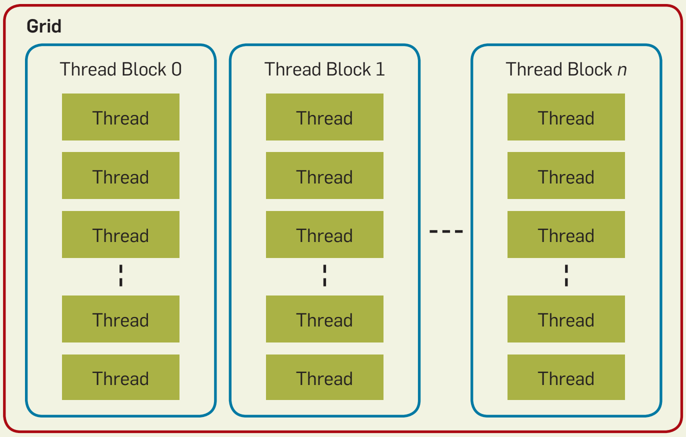


Figure 4. Multiple independent audio streams calculated in parallel.

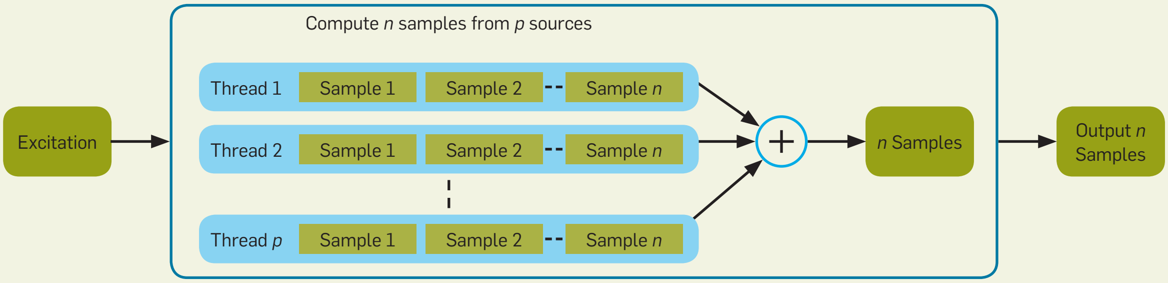
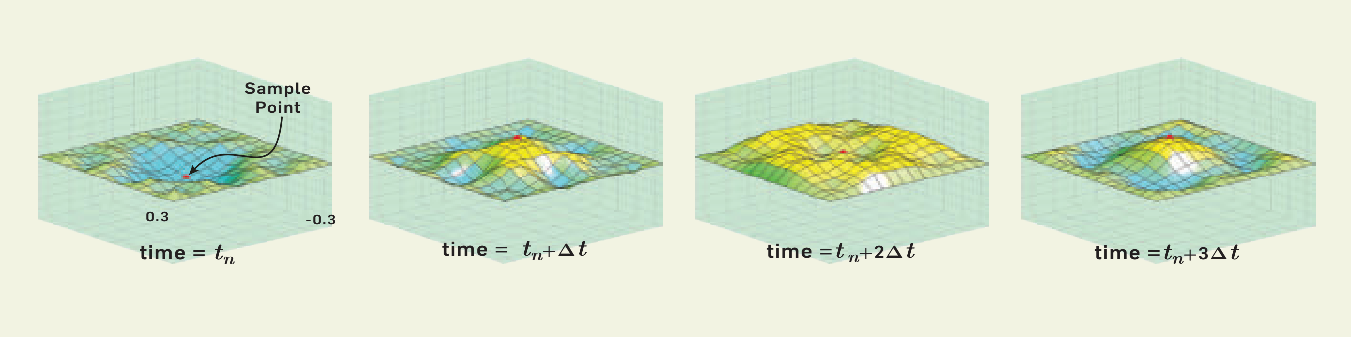


Figure 5. Vertical displacements of points on flat plate after being struck.



Finite Difference Approximation

Physical objects are frequently modeled using differential equations. To perform numerical simulations of these objects, finite difference approximations of the differential equations are commonly used. For our work, we use an approximation of the *2D wave equation*, which describes vibrations in two dimensions through an object. Consider exciting a flat rectangular plate to produce sound. The plate is modeled as a horizontal 2D grid of points. When the plate is struck, points in the grid "bounce" up and down very fast, resulting in vibration and sound, as shown in Figure 5.

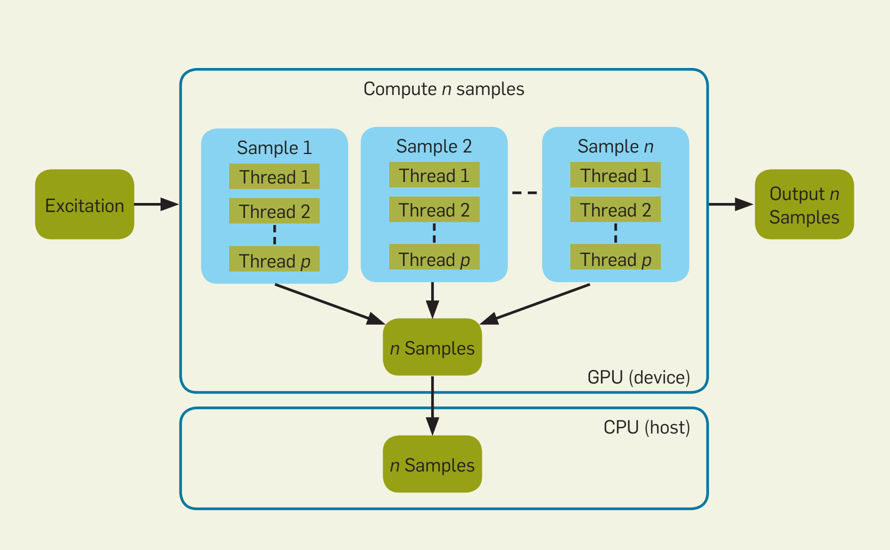
In the simulation, 2D arrays keep track of the vertical displacement of the plate at each point. One array stores the current displacements, while arrays of the two previous time steps are retained. To calculate the displacement of a point at the current time step, previous displacement values around the point being calculated are used. Suppose $x_{i,j}(t)$ contains the vertical displacement at time t of the point at (i,j). In a previous paper,[2] we saw that $x_{i,j}(t+1)$ can be calculated from $x_{i,j}(t)$, the four nearest neighbors of $x_{i,j}$ at time t, and $x_{i,j}(t-1)$. A *sample point* at (for example) the center of the grid, marked in red in Figure 5, can be monitored to produce the audio samples for the output sound.

In a straightforward finite difference implementation, computing a $W \times W$ grid of data points at time t requires W^2 steps, and depends on the $W \times W$ grid at the previous two time steps.

Challenges of Implementing the Finite Difference Technique

Using a finite difference-based simulation to generate a single stream of audio is a compute-intensive endeavor. Compared with the calculation of one output sample from a few sinusoidal oscillators or filters, generating a single sample in a finite difference simulation involves significantly more arithmetic and memory operations. Hence, the computation of a single sample in real time has to be spread over multiple threads. Figure 6 shows a high-level view of a GPU-based finite difference simulation.

Figure 6. Real-time finite difference audio synthesis.



We faced three major challenges implementing a real-time finite difference synthesizer on the GPU. First, *kernel launch overhead*, a delay from the time the host executes the kernel on the device until the device begins execution of the kernel, may be significant. Second, the limit on the number of available threads per block restricted how the simulation grid was mapped onto the GPU. Third, the inability to synchronize or order block execution limited the way blocks of threads could be configured and executed on the GPU device.

Figure 7 shows two types of parallel audio applications: Figure 7a demonstrates multiple independent audio streams, while 7b shows parallel finite difference simulation. As mentioned previously, with independent-stream audio processing it is usually feasible to configure the system as shown in Figure 7a; each thread freely computes an independent stream of *n* samples simultaneously, producing a buffer of *n* audio samples at the end of a period of computation. A single synchronization event occurs after *n* time steps, waiting for all threads to complete their calculations. After the synchronization event, these buffers of audio data are then organized before being sent back to the host. For example, multiple sources may be mixed together or organized to maintain temporal coherence.

Suppose $x(t)$ is a two-dimensional array of vertical displacements at time t. With finite difference simulations, recall that to calculate vertical displacement of a point at i,j at time $t+1$, you need to refer to the point and its nearest neighbors at time t and `t`–`1`; these calculations need to be performed over the entire `x` array. To generate a buffer of audio samples over *n* time steps, you capture the vertical displacement of a single sample point at the same location in `x` over time; to do this it is necessary to calculate the `x(`*t*`+1)` to `x(`*t*`+`*n*`)` simulation arrays, while building a buffer of *n* samples of vertical displacement from the sample point in `x`. At time *t*`+1`, it is necessary only to retain the arrays for times *t* and *t*–`1`. Pseudocode for a sequential finite difference simulation is shown in Figure 8.

As shown in Figure 6, you specify *p* = `W`×`W` threads to execute the GPU kernel. You spread the inner-loop calculation of the `x(`*t*`+1)` array over multiple threads. Each thread computes one point at `(myRow, myColumn)` of `x(`*t*`+1)`, with `myRow` and `myColumn` based on the thread's unique ID. The parallelized pseudocode is shown in Figure 9.

To calculate a buffer of audio samples over *n* time steps, you simply make one kernel call. Since the calculations at `x(`*t*`+1)` depend upon the completed calculations of the two previous time steps `x(`*t*`)`, `x(`*t*–`1)`, you must synchronize after each time step (Figure 7b). The time spent in the *n* synchronizations in the kernel is critical to the efficiency of this approach. CUDA provides fast hardware synchronization support but only for threads within the same block. Therefore, all calculations must be performed using threads within a *single* block (Figure 7b). Using a single block allows you to synchronize using fast mechanisms native to CUDA, but you can no longer leverage the efficiency of allowing the GPU to schedule multiple blocks of threads. Since there is only one block of threads, only one SM can operate on one finite difference approximation at any time. You could, however, simulate more than one finite difference-based instrument simultaneously, up to the number of SMs on the device.

To use multiple blocks of threads and multiple SMs, you can try configuring the kernel to calculate one time step of `x(`*t*`)` at a time, returning control to the host after each time step. The synchronization is taken care of in the return from the kernel. In this solution, the configuration of blocks of threads depends on the locality of the data being accessed and calculated; it is a standard GPU optimization problem. However, this approach has a problem very similar to the naïve audio generation problem described previously in Figure 1a; there is an overhead to the host executing a kernel on the device, and the device returning from the kernel. This kernel launch overhead builds linearly and is not insignificant. Our experiments have shown that on an NVIDIA GeForce GT 650M, on average, the minimum time to execute and return from a kernel is around 17μs, with some initial delays of 2ms or longer. Recall that CD-quality audio requires generating one sample per 23μs. This means that even with the minimum overhead, there are about 6μs to calculate a sample in real time. This is unrealistic for finite difference calculations.

Hence, we took the approach of using a single block of threads, with *n* time steps per kernel call, as shown in the previous pseudocode. However, we ran into another constraint: the maximum number of threads in a block is fixed in each GPU implementation and depends further on hardware resource constraints such as the number of registers and SPs. For larger simulation grids with more points than threads per block, each thread must be able to calculate a rectangular *tile* of several points. For example, the GeForce GT 650M supports up to 1,024, or 32×32 threads per block. To simulate a 64×64 grid, each thread would calculate a tile of 2×2 points.

Figure 7. Two types of parallel audio applications.

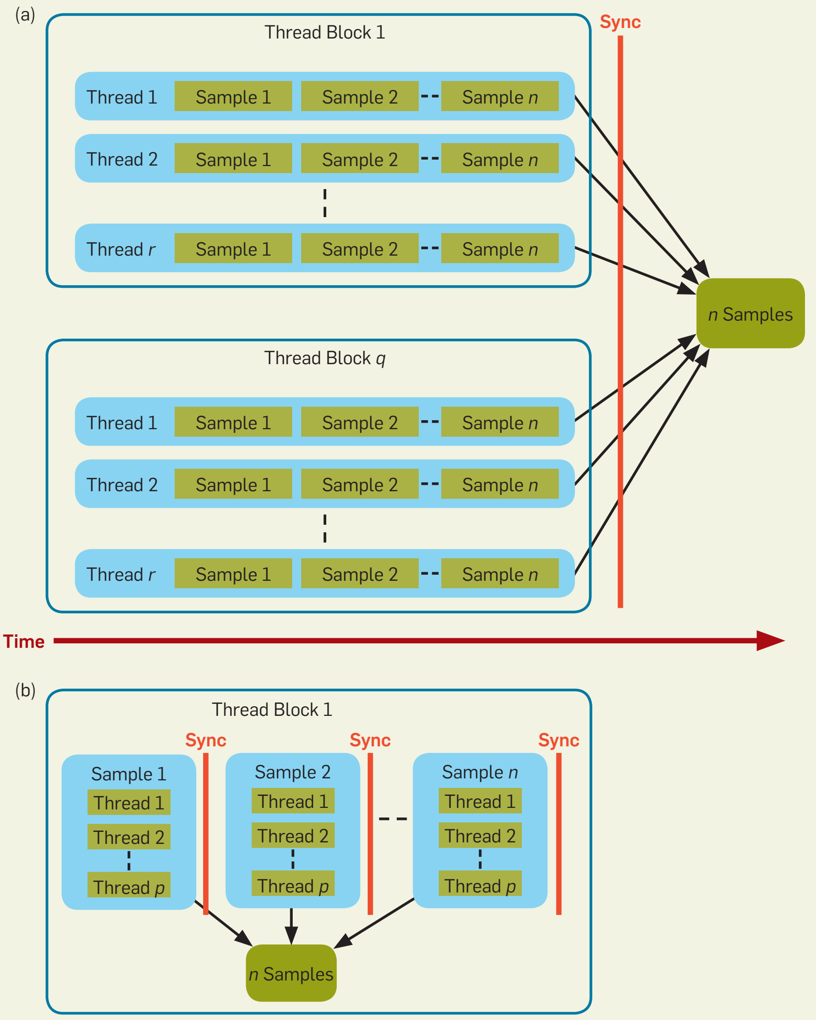


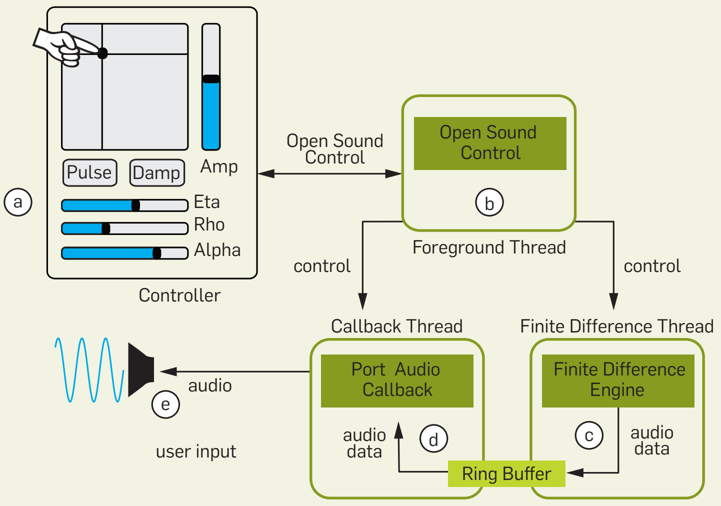
Figure 8. Pseudocode for a sequential finite difference simulation.

```
// x1ptr: ptr to x(t+1) array
// x0ptr: ptr to x(t) array
// xmptr: ptr to x(t-1) array
// buffer: n-sample audio output buffer
// W: width of plate
for bufferIndex 0 to n-1
   for row 0 to W-1
      for column 0 to W-1
         Calculate x1ptr[row][column] using x0ptr, xmptr
      End for column
   End for row

   Write sample point to buffer[bufferIndex]
   tempPtr = xmptr;
   xmptr = x0ptr;
   x0ptr = x1ptr;
   x1ptr = tempPtr;

End for bufferIndex
```

Figure 9. Pseudocode for a parallelized finite difference simulation.

```
for bufferIndex 0 to n-1

   Get myRow, myColumn from GPU thread ID
   Calculate x1ptr[myRow][myColumn] using x0ptr, xmptr

   Synchronize threads

   if [myRow][myColumn] is the sample point
      Write x1ptr[myRow][myColumn] to buffer[bufferIndex]

   tempPtr = xmptr;
   xmptr = x0ptr;
   x0ptr = x1ptr;
   x1ptr = tempPtr;

End for bufferIndex
```

Figure 10. Finite difference synthesizer program structure.



The Finite Difference Synthesizer

Our software synthesis package, Finite Difference Synthesizer (FDS), was designed to operate on Mac OS X and Linux. FDS simulates a vibrating plate, similar to a percussion instrument. The system (Figure 10) has three primary components: the controller interface (Figure 10b), the finite difference engine (Figure 10c), and the audio callback handler (Figure 10d). Each of these three components runs in its own thread.

The controller interface (Figure 10b) is the program's foreground thread. It includes a listener loop that receives control and configuration messages from an external controller, via the Open Sound Control (OSC) protocol (http://opensoundcontrol.org).

To use FDS, a performer manipulates an external OSC-capable controller (Figure 10a), which may be a keyboard, drum pad, or tablet. An OSC message is sent from the controller to FDS's foreground thread. This message may change settings (simulation parameters, strike location, among others) or trigger an excitation event (strike the plate, damp it, and so on). The thread then initiates the corresponding operations in the finite difference and audio callback threads.

To address the implementation challenges previously described, we created a *finite difference engine* (Figure 10c). This engine runs continually in its own thread, executing the finite difference simulation and generating audio data. This engine thread is the only one in FDS that interacts with the GPU device. It contains a control loop running on the host, keeping the finite difference simulation running on the device (Figure 11). The control loop on the host part of the engine waits for control signals from the foreground (control) thread, such as excitation and damping events. When an excitation event is received, the host adds a precalculated 2D Gaussian impulse of vertical displacements into the finite difference grid, maintaining the current waveform while adding the energy from the excitation event. The center of the impulse is determined by the location of the "strike" on the grid (Figure 5).

Audio data is transferred from the device to the host using memory shared by both the host and device. This eliminates one of the major bottlenecks formerly associated with GPUs: the device-host memory transfers. The host copies the audio data to a ring buffer shared with the *audio callback thread*, which handles getting audio data to the audio driver.

The audio callback thread communicates with PortAudio (http://www.portaudio.com), a cross-platform audio driver that coordinates the interface between FDS and the operating system's audio layer. When the PortAudio driver is ready for more audio data, it executes a callback function in the audio callback thread. This function copies data placed in the ring buffer by the finite difference thread to the Port Audio output buffer. The output buffer is then sent to the operating system for playback.

Results

To evaluate whether FDS constitutes a useful development in software synthesis, we asked two related questions: Is FDS able to generate real-time audio based on finite difference simulations, for an interesting range of simulation parameters, at reasonable latencies? How does FDS's performance on a GPU compare with a single-threaded finite difference simulation executed on a CPU, with identical simulation parameters? For the second question, note that GPUs and CPUs have very different architectures; different systems can have very different CPU/GPU combinations in terms of performance. Hence, our CPU-vs.-GPU comparisons should be considered *practical* references for application end users; they are not intended as rigorous comparative performance studies. Even for GPU-to-GPU comparisons, the models vary widely in their capabilities and system implementations, making these comparisons difficult.

For our measurements, we kept the audio buffer size constant and ran finite difference simulations for a number of simulation grid sizes on both CPUs and GPUs. Large grid sizes are important for generating sounds with low pitches and simulations with high spatial resolution. We also monitored the audio output buffer for underruns (that is, when audio data is not being produced fast enough to keep up with the demands of audio playback). This produces gaps in the audio output data, which are audible as glitches or other unpleasant artifacts as the audio system waits for data to be ready.

The results of these experiments are obviously highly system dependent. We have implemented versions of FDS on GPUs for some years. On the earlier platforms, we were able to execute FDS for up to 21×21 simulation grids in real time on an NVIDIA GTX285 GPU, with audio buffer size of 4,096 samples, without audio buffer underruns; on the 3GHz Intel Xeon 5160 CPU on the same system, audio buffer underruns were reported for all but trivially small grid sizes.[2]

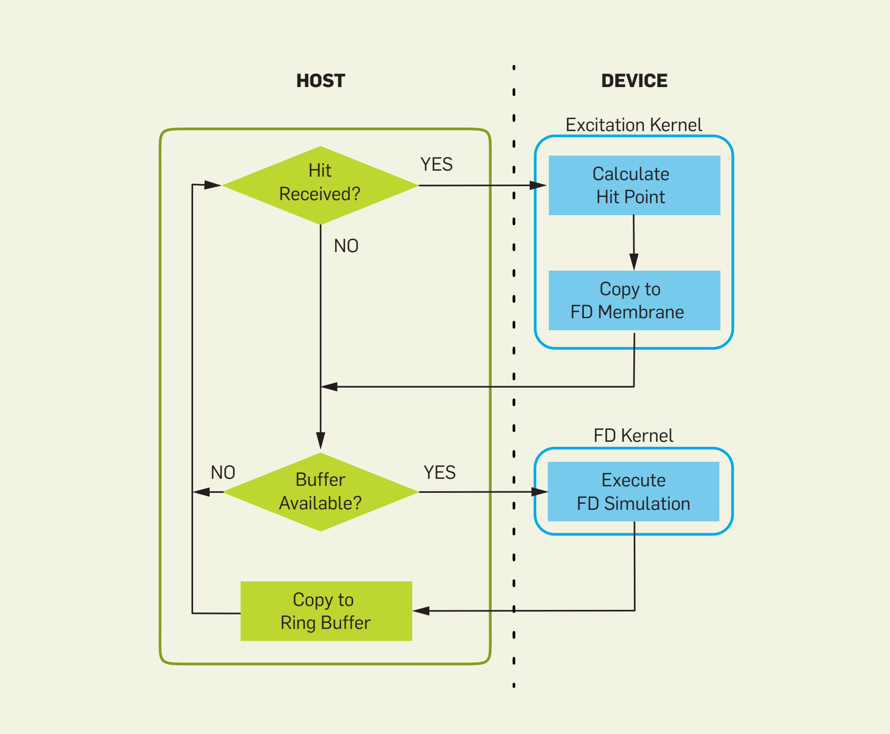
Our latest test platform is a mid-2012 MacBook Pro Retina, with a 2.7GHz Intel Core i7 processor and 16GB of RAM. This system has a built-in 900MHz NVIDIA GeForce GT 650M GPU, which has two SMs with 192 SPs each. The 650M is an implementation of NVIDIA's latest Kepler architecture, optimized for power efficiency; hence, while it has numerous enhancements over the earlier Tesla and Fermi architectures, the 650M is one of the slower Kepler-based GPUs. The operating system is OS X version 10.8.2, running CUDA 5.0. We timed the finite difference kernel execution within the FDS program infrastructure, taking measurements around the kernel calls.

Figure 12 shows the time needed to generate one 512-sample buffer of audio for FDS running on the CPU and GPU of the MacBook Pro Retina. Execution times above 11ms produce audio buffer underruns and cannot be used for audio playback. These numbers are included for speed comparison only.

For our tests we kept the audio output buffer size at 512 samples. This means that FDS needed to produce and have ready 512 samples every 23µs × 512 = 11.61 ms. Simulations with execution times above 11ms produce buffer underruns and are unusable. We were able to obtain good results—meaning audio playback with no buffer underruns—for grid sizes as large as 69×69 on the CPU, and grid sizes as large as 84×84 for the GPU, or a 48% improvement in the maximum grid size supported.

As mentioned earlier, performance analysis involving GPUs and CPUs is tricky. At the risk of comparing even more apples and oranges, we introduce another point of reference. Figure 13 shows measurements made on a system with a 2GHz Intel Xeon E5504 CPU and a 1.15GHz NVIDIA Tesla C2050

Figure 11. Finite difference engine thread.



GPU, designed as a GPU-based server for scientific applications. The C2050 is based on NVIDIA's Fermi architecture; this implementation is targeted at the high-performance computing market, with power consumption being a lower priority. For these measurements, we again timed the finite difference kernel execution independent of the FDS program infrastructure, which does not run on the GPU. The audio output buffer size was again 512 samples. The C2050 supports simulation grid sizes up to 81×81, but the largest grid size that the slower Xeon CPU can support is about 27×27.

Summary and Future Work

Our experiments have shown it is possible to run finite difference-based simulations on the GPU to generate real-time audio with reasonable latency, and for grid sizes larger than is possible on a CPU. We note that CPU-vs.-GPU comparisons are tricky at best; our latest measurements were made on the GPU and CPU for the mid-2012 MacBook Pro Retina. We were surprised that the Kepler-based 650M, designed for power efficiency, had comparable performance on FDS to the slightly earlier Fermi-based C2050, which was designed for high-performance computing. (This is partly because a single finite difference simulation can use only one SM at a time; the 650M has two SMs, while the C2050 has 14, so a much larger fraction of the C2050's hardware resources is idle). We were also surprised that current CPUs such as the Intel Core i7 exhibit competitive performance at medium grid sizes; an obvious future direction is to port FDS to multicore CPU systems for comparison with GPUs.

Our current implementation of the finite difference approximation on the GPU is straightforward. In addition, we plan to study approaches to optimize the software for larger grid sizes, for multiple finite difference-based instruments, and to support different simulation geometries. We also plan to port FDS to other GPU computing platforms such as OpenCL for testing on other GPU architectures.

Figure 12. Time to generate audio buffer with FDS on a mid-2012 MacBook Pro Retina.

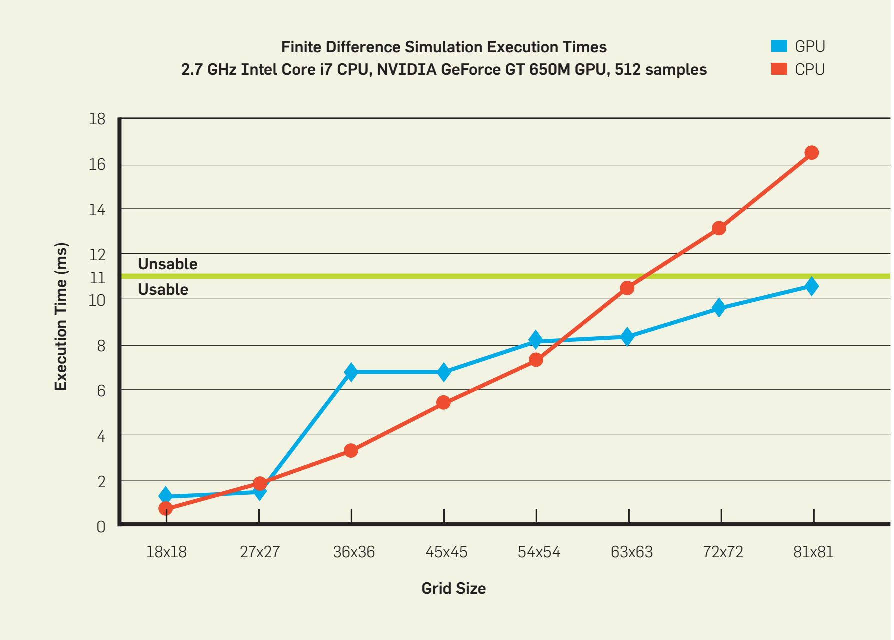
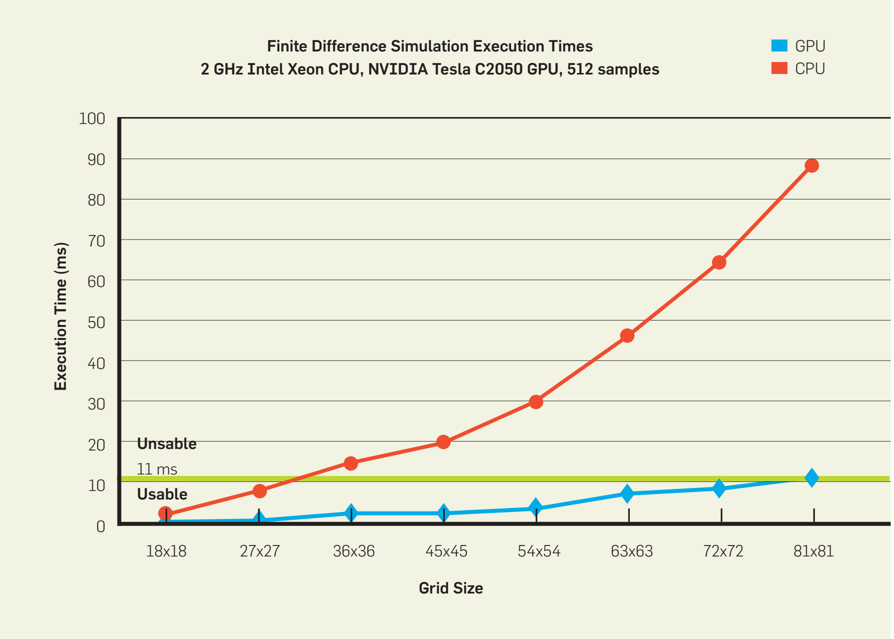


Figure 13. Time to generate audio buffer with FDS on a GPU server.



Related articles on queue.acm.org

GPUs: A Closer Look
Kayvon Fatahalian, Mike Houston
http://queue.acm.org/detail.cfm?id=1365498

Computing without Processors
Satnam Singh
http://queue.acm.org/detail.cfm?id=2000516

Real-time Computer Vision with OpenCV
Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, Victor Eruhimov
http://queue.acm.org/detail.cfm?id=2206309

References

1. Blibao, S. *Numerical Sound Synthesis: Finite Difference Schemes and Simulation in Musical Acoustics.* John Wiley and Sons, Chichester, UK, 2009.
2. Sosnick, M. and Hsu, W. Efficient finite difference-based sound synthesis using GPUs. In *Proceedings of the Sound and Music Computing Conference* (Barcelona, Spain, July 2010).
3. Zhang, Q., Ye, L. and Pan, Z. Physically based sound synthesis on GPUs. In *Entertainment Computing-ICEC 2005, Lecture Notes in Computer Science*, Springer, Berlin, 2005, 328–333.

Bill Hsu is an associate professor of computer science at San Francisco State University. His current interests include high-performance computing, audio analysis and synthesis, physics-based modeling, and audiovisual performance systems.

Marc Sosnick-Pérez is a graduate student, teacher, and researcher at San Francisco State University. He is part of SFSU's SETAP (Software Engineering Team Assessment and Prediction) project exploring novel means of assessing student learning, and he was part of the CAMPAIGN project exploring the use of GPU-accelerated clustering algorithms.

DOI:10.1145/2461256.2461274

Article development led by acmqueue
queue.acm.org

Building a distributed system requires a methodical approach to requirements.

BY MARK CAVAGE

There Is No Getting Around It: You Are Building a Distributed System

DISTRIBUTED SYSTEMS ARE difficult to understand, design, build, and operate. They introduce exponentially more variables into a design than a single machine does, making the root cause of an application problem much harder to discover. It should be said that if an application does not have meaningful service-level agreements (SLAs) and can tolerate extended downtime and/or performance degradation, then the barrier to entry is greatly reduced. Most modern applications, however, have an expectation of resiliency from their users, and SLAs are typically measured by "the number of nines" (for example, 99.9 or 99.99% availability per month). Each additional nine becomes more difficult to achieve.

To complicate matters further, it is extremely common that distributed failures will manifest as intermittent errors or decreased performance (commonly known as brownouts). These failure modes are much more time consuming to diagnose than a complete failure. For example, Joyent operates several distributed systems as part of its cloud-computing infrastructure. In one such system—a highly available, distributed key/value store—Joyent recently experienced transient application timeouts. For most users the system operated normally and responded within the bounds of its latency SLA. However, 5%–10% of requests would routinely exceed a predefined application timeout. The failures were not reproducible in development or test environments, and they would often "go away" for minutes to hours at a time. Troubleshooting this problem to root cause required extensive system analysis of the data-storage API (node.js), an RDBMS (relational database management system) used internally by the system (PostgreSQL), the operating system, and the end-user application reliant on the key/value system. Ultimately, the root problem was in application semantics that caused excessive locking, but determining root cause required considerable data gathering and correlation, and consumed many working hours of time among engineers with differing areas of expertise.

The entire preamble is fairly common knowledge to engineers working on distributed systems, and the example application degradation is typical of a problem that arises when operating a large system. Current computing trends, however, are bringing many more organizations and applications into the distributed-computing realm than was the case only a few years ago. There are many reasons why an organization would need to build a distributed system, but here are two examples:

- The demands of a consumer website/API or multitenant enterprise application simply exceed the computing capacity of any one machine.
- An enterprise moves an existing application, such as a three-tier system, onto a cloud service provider in order to save on hardware/data-center costs.

The first example is typical of almost any Internet-facing company today. This has been well discussed for many years and is certainly a hotbed for distributed-systems research and innovation, and yet it is not a solved problem. Any application going live still needs to assess its own requirements and desired SLAs, and design accordingly to suit its system; there is no blueprint. The usage patterns of any new successful applications will likely be different from those that came before, requiring, at best, correct application of known scaling techniques and, at worst, completely new solutions to their needs.

The second example is increasingly common, given current trends to move existing business applications into the cloud to save on data-center costs, development time, etc. In many cases, these existing applications have been running on single-purpose systems in isolated or low-utilization environments. Simply dropping them into an environment that is often saturated induces failure more often than these applications have ever seen or were designed for. In these cases, the dirty secret is that applications must be rewritten when moving to cloud environments; to design around the environment in which an application will be running requires that it be built as a distributed system.

These two examples are at opposite extremes in terms of the reasons why they require a distributed solution, yet they force system designers to address the same problems. Making matters worse, most real-world distributed systems that are successful are built largely by practitioners who have gone through extensive academic tutelage or simply "come up hard," as it were. While much of the discussion, literature, and presentations focus on the en vogue technologies such as Paxos,[3] Dynamo,[5] or MapReduce[4]—all of which indeed merit such focus—the reality of building a large distributed system is that many "traditional" problems are left unsolved by off-the-shelf solutions, whether commercial or open source.

This article takes a brief walk through the reality of building a real-world distributed system. The intent is neither to provide a toolkit, or any specific solutions to any specific problems, nor to dissuade new builders from constructing and operating distributed systems. Instead, the goal is simply to highlight the realities and look at the basics required to build one. This is motivated by a trend toward building such systems on "hope and prayer." Too many applications are being built by stringing together a distributed database system with some form of application and hoping for the best. Instead—as with all forms of engineering—a methodical, data-driven

Too many applications are being built by stringing together a distributed database system with some form of application and hoping for the best. Instead—as with all forms of engineering—a methodical, data-driven approach is needed.

approach is needed, and the goal of this article is to provide useful tips on using such a methodology.

While it is important to disambiguate "the cloud" from a distributed system, for practical reasons most new distributed systems are being built in a cloud—usually a public one—and so the rest of this article assumes that environment. Furthermore, this article focuses specifically on online Web services (what once upon a time was referred to as OLTP—online transaction processing—applications), as opposed to batch processing, information retrieval, among others, which belong in an entirely separate problem domain. Finally, this article leverages a hypothetical—but not purely imagined—application in order to provide concrete examples and guidance at each stage of the development life cycle.

Architecting a Distributed System

One of the most common terms that applies to a distributed-system architecture is service-oriented architecture (SOA). While SOA may conjure (to some readers) unpleasant flashbacks of CORBA brokers and WS-* standards, the core ideas of loosely coupled services—each serving a small function and working independent from one another—are solid, and they are the foundation of a resilient distributed system. Drawing parallels to a previous generation, services are the new process; they are the correct abstraction level for any discrete functionality in a system.

The first step in constructing a service-oriented architecture is to identify each of the functions that comprise the overall business goals of the application and map these into discrete services that have independent fault boundaries, scaling domains, and data workload. For each of the services identified, you must consider the following items:

- *Geographies.* Will this system be global, or will it run in "silos" per region?
- *Data segregation.* Will this system offer a single- or multi-tenancy model?
- *SLAs.* Availability, latency, throughput, consistency, and durability guarantees must all be defined.
- *Security.* IAAA (identity, authentication, authorization, and audit), data confidentiality, and privacy must all be considered.
- *Usage tracking.* Understanding usage of the system is necessary for at minimum day-to-day operations of the system, as well as capacity planning.[1] It may also be used to perform billing for use of the system and/or governance (quota/rate limits).
- *Deployment and configuration management.* How will updates to the system be deployed?

This is not a comprehensive list of what it takes to operate a distributed system (and certainly not a list of requirements for running a business based on a distributed system, which brings a lot more complexity); rather it is a list of items to be addressed as you define the services in a distributed system. Additionally, note that all the "standard" software-engineering problems, such as versioning, upgrades, and support, must all still be accounted for, but here the focus is purely on the aspects that are core (but not necessarily unique) to a given service in a distributed system.

Consider the following hypothetical application: a Web service that simply resizes images. On the surface this appears to be a trivial problem, as image resizing should be a stateless affair, and simply a matter of asking an existing operating system and library or command to perform an image manipulation. However, even this very basic service has a plethora of pitfalls if your desire is to operate it at any scale. While the business value of such a service may be dubious, it will suit our purposes here of modeling what a distributed system around this utility would look like, as there is enough complexity to warrant quite a few moving parts—and it only gets harder the more complex the service.

Example System: Image-Resizing Service

Let's define the characteristics of the example Web service from a business point of view. Since the service will simply resize images, the system will need to ingest images, convert them, and make them available to the customer as quickly as possible—with a response time that is acceptable to an interactive user agent (that is, Web browser). Given the resizing is to be done in real time, and to simplify this article, let's assume the images are immediately downloaded and their long-term storage is not within the confines of this system. Customers will need to be able to securely identify themselves (the scheme by which they do so is outside the scope of this article, but we can assume they have some way of mapping credentials to identity). The service should offer a usage-based pricing model, so it needs to keep track of several dimensions of each image (for each request), such as the size of image and the amount of CPU time required to convert it. Finally, let's assume each data center where the system operates is isolated from all others, with no cross-data-center communication.

In terms of adoption, let's just throw a dart at the wall and say the number of users is to be capped at 100,000 and the number of requests per second is to be 10,000 in a single region (but this number will be available in each region). The business needs the system to provide 99.9% availability in a month, and the 99th percentile of latency should be less than 500ms for images less than one megabyte in size.

While this is a contrived example with an extremely short list of requirements—indeed, a real-world system would have substantially better-qualified definitions—we can already identify several distinct functions of this system and map them into discrete services.

Break Down Into Services

As stated earlier, the initial step is to decompose the "business system" into discrete services. At first glance, it may seem an image-resizing service needs only a few machines and some low-bar way to distribute resizes among them. Before going any further, let's do some back-of-the-envelope math as to whether this holds true or not.

Recall the goal is to support 10,000 resizes per second. For the purposes of this article, let's limit the range of input formats and define an average input size to be 256KB per image, so the system can process 10 conversions per second per CPU core. Given modern CPU architectures, we can select a system that has 32 cores (and enough

DRAM/IOPS capacity so there is no worry about hitting other boundaries for the purpose of this example). Obviously such a system can support 320 conversions per second, so to support the goals of 10,000 images per second, the business needs a fleet of at least 32 image-processing servers to meet that number alone. To maintain 20% headroom for surges, an extra seven machines is required. To point at a nice even number (and since this is notepad math anyway), the ballpark figure would be 40 systems to meet these goals. As the business grows, or if the average conversion time goes up, it will need more systems, so assume that 40 is a lower bound. The point is, this image service needs enough systems that it requires a real distributed stack to run it.

Now that we have confirmed a distributed system is necessary, the high-level image requirements can be mapped into discrete services. I will first propose an architecture based on my experience building distributed systems, though there are many alternative ways this could be done. The justification for the choices will follow this initial outlay; in reality, arriving at such an architecture would be much more of an iterative process.

The diagram in the accompanying figure breaks down the system into a few coarse services:

- A customer-facing API (such as REST, or representational state transfer, servers).
- A distributed message queue for sending requests to compute servers.
- An authoritative identity-management system, coupled to a high-performance authentication "cache."
- A usage aggregation fleet for operations, billing, and so on.

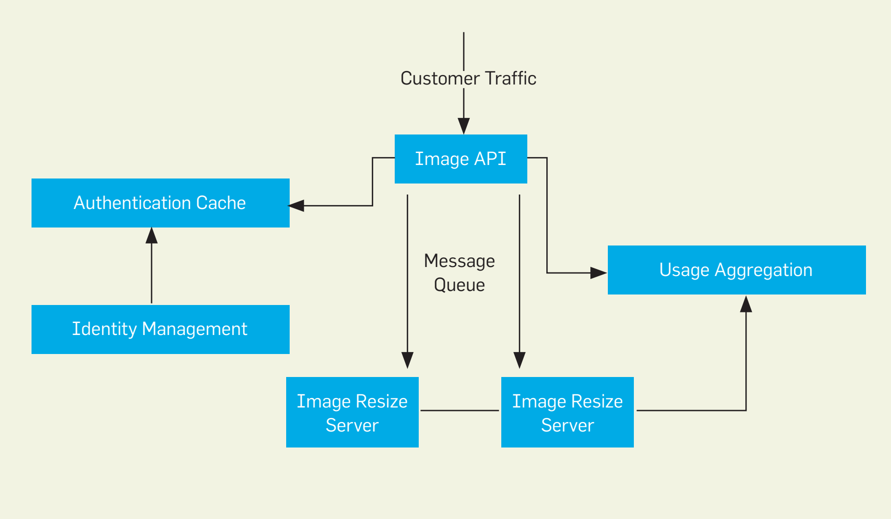
There are, of course, other ways of decomposing the system, but the above example is a good starting point; in particular, it provides a separation of concerns for failure modes, scaling, and consistency, which we will elaborate on as we iterate through the components.

Image API Details

Starting from the outside in, let's examine the image API tier in detail. First, a key point must be made: the API servers should be stateless. You should strive to make as many pieces in a distributed system as stateless as possible. State is always where the most difficult problems in all aspects arise, so if you can avoid keeping state, you should.

- *Geographies.* As the requirements are to operate in a single region, the API server(s) offers access to the system only for a single region.
- *Data segregation.* API servers are stateless, but it is worth pointing out that to deliver 10,000 resizes per second across a large set of customers, the system must be multitenant. The variance at any given time across customers means we would be unable to build any dedicated systems.
- *SLA availability.* Availability is the worst number either the software/systems or the data center/networking can offer. Since the business wants 99.9% availability from the system, let's set a goal for this tier at 99.99% availability in a given month, which allows for 4.5 minutes of downtime per month. This number is measured from the edge of the data center, since the system cannot control client network connectivity.
- *SLA latency.* The latency of any request in the API server is the sum of the latencies to each of the dependent systems. As such, it must be as low as possible: single-digit milliseconds to (small) tens of milliseconds at the 99th percentile.
- *SLA throughput.* As defined earlier, the system must support 10,000 requests per second. Given a non-native implementation, we should be able to support 5,000 requests per second on a given server (this number is drawn purely from practical experience). Keeping with a goal of 20% headroom, the system needs three servers to handle this traffic.
- *SLA consistency and durability.* The API tier is stateless.
- *IAAA.* Each customer needs his or her own identity/credentials. Users must not be able to access each other's input or output images (recall we are not offering long-term storage, where sharing would be meaningful). A record of each resize request must be saved.
- *Usage tracking.* We want to "meter" all requests on a per-customer basis, along with the request parameters—when implementing the API, we need to ensure the system captures all dimensions about the request to be passed along.
- *Deployment and configuration management.* A goal for system operations is to manage all deployments and configuration of the system in one manner. We can table this for now and touch on it later on.

The distributed services of an image resize service.



We have established the need for more than one API server, so load balancing of traffic must be managed across all API servers. Since there is ample prior art[10] and known solutions to the problem, for this particular aspect I will be brief and simply point out that it must be addressed. Solutions typically involve some combination of round-robin DNS (Domain Name System) records to public IP addresses, where these IP addresses are greater than one load balancer resolved by protocols such as CARP (Common Address Redundancy Protocol) or similar solutions involving IP address takeovers at L2/L3 (layer 2/layer 3).

Finally, it is worth pointing out the API tier will certainly need a custom implementation. The semantics of this are so tied to the business that in the build-or-buy trade-off, there is really no choice but to build. Load balancing, however, offers many open source and commercial choices, so this is a good place to start. Specialized aspects of the API may require us to build our own, but it would not be prudent to start by assuming so.

It is worth pointing out the API tier will certainly need a custom implementation. The semantics of this are so tied to the business that in the build-or-buy trade-off, there is really no choice but to build.

Messaging

There is a discrepancy in scale between the API servers and the image-processing servers, thus requiring a scheduling mechanism to deliver images to available transcoding servers. There are many ways to achieve this, but given the individual systems can schedule their own cores, a relatively obvious starting point is to use a distributed message queue. A message queue could "stage" images locally on the API server that took the user request and push a ready message onto the queue. This has the nice property of providing a first-pass scheduling mechanism—FIFO—and allows graceful latency degradation of user resize times when the system is too busy. It should be stated again, however, that as is true of all components in a distributed system, message queues are not without trade-offs. In particular, message queues typically are often viewed as both highly available and consistent, but as the CAP theorem (meaning consistency, availabilit,y and partition tolerance) says, we cannot have our cake and eat it too.

Generally speaking, message queues prioritize consistency over availability; most open source and commercial message queues are going to prioritize consistency and run in some nature of active/standby mode, where the standby mode gets a (hopefully) synchronous copy of messages that are pushed and popped from the queue. While this is likely the right common choice, it does mean the example system must automate downtime, as promotion of a standby and updating replication topologies are nontrivial. Before discussing what that solution would look like, let's run through the requirements again:

- *Geographies.* This is not applicable. We will run a message queue per deployment.
- *Data segregation.* Multitenant—there is one logical message queue per deployment.
- *SLA availability.* The system needs to provide 99.9% availability, and since we are assuming we will not be writing a new message queue from scratch, we need to automate failover to minimize this number as much as possible.
- *SLA latency.* Latency in a non-queued case (obviously if the consumer is reading slowly, latency goes up) needs to be as small as possible. The goal is sub-millisecond latency on average, and certainly single-digit millisecond latency for the 99th percentile of requests. Luckily, most message queues are capable of providing this, given they are in-memory.
- *SLA throughput.* Because we are using the messaging queue bidirectionally, the system needs to support 20,000 requests per second (24,000 with 20% headroom). The number of systems needed to support this is dependent on the choice of message-queue implementation.
- *SLA consistency and durability.* As previously stated, the queue must guarantee FIFO semantics, so we must have a strong consistency guarantee.
- *IAAA.* This is not applicable.
- *Usage tracking.* While we are interested in operational metrics, this is not applicable.
- *Deployment and configuration management.* Failovers must be automated to meet the availability SLAs.

Automating Failover

To automate failovers, the system needs an extra mechanism in place to perform leader election. While there is much literature on the subject of leader election, and many algorithms have come in over the years, we would like to have a distributed consensus protocol (for example, Paxos) in place. Ultimately, we need to perform leader election and failover reliably, and the system must rely on an authoritative system regardless of the network state. Algorithms such as Paxos guarantee all parties will reach the same state, and as long as the infrastructure that actually mucks with the topology relies on the consensus protocol, the correct ordering of topology changes is guaranteed, even though they may take longer than desired (distributed consensus protocols cannot guarantee when parties reach a value, just that they will, eventually). While the system may experience longer-than-desired outage times until all parties get the right value, the time (hopefully) will be less than it would be had we relied on manual operations to manage this process.

Note that in practice this is not an easy undertaking; many experienced engineers/administrators are of the opinion you are better off having manual failover and living with the increased outage times when it does happen, as this process has to be perfect or risk worse problems (for example, data corruption, incorrect topologies, and so on). Additionally, many message queues and database systems do provide guidance and some mechanism for failover that may be good enough, but most of these systems do not (and cannot) guarantee correct topologies in true network partitions and may result in "split-brain" scenarios. Therefore, if your system must support a tight availability SLA and you need strong consistency, there is not much choice in the matter.

Thankfully, we have designed the example system in such a way the actual conversion servers are stand-alone (modulo the message queue). As such, there is not anything to cover with them that is not covered elsewhere.

Platform Components

In the system as described here, only a subset of the requirements is actually supporting the core problem definition of image resize. In addition to the image requirements, the system must support authentication/authorization and usage collection (many more components would be needed to run a business, but for the purposes of this article I will cap it here). Also, to avoid repetition, this section will touch only on the highlights of why the architecture looks like it does rather than continue to step through each of the categories.

> **Whether you are able to leverage existing components or build your own, all of them will need to consider how to manage production deployment at scale.**

Identity and authentication. The example service needs an identity-management system that supports authentication, authorization, and user management. First, the business can identify that customer management is going to occur far less often than Web service requests (if not, it is out of business). So the business hopes to serve orders-of-magnitude more authentication requests than management requests. Additionally, since this is a critical function but does not add value to the actual service, latency should be as low as possible. Given these two statements, we can see that scaling the authentication system is necessary to keep up with service requests—authentication is, generally speaking, not cacheable if done correctly.

While authentication events themselves are not cacheable, however, the data required to serve an authentication request is, and so we can still treat this system as largely read-only. This observation allows the decoupling of the "data plane" from the "management plane" and scales them independently. The trade-off is you end up with two separate systems, and in any such case, consistency between them will, by necessity, be asynchronous/eventual. In practical terms this means introducing delay between customer creation/update and the availability of this data in the data plane. This technical decision (the A in CAP) does impact the business, as new/updated customers may not be able to use the system right away. The system may need to maintain an aggressive SLA for data replication, depending on business requirements.

Usage collection. Usage collection is the direct inverse problem of authentication; it is highly write intensive, and the data is read infrequently—often only once—usually to process billing information (business analytics such as data warehousing are outside the scope of this article). Additionally, consistency of metering data is not important, as the only consumer is (typically) internal tools. Some important trade-offs can be made regarding durability; any single record or grouping of records is not valuable until aggregated into a final customer-visible accounting. You can therefore separate durability SLAs appropriately. The difficult aspect of metering is the need to process a larger number of records than requests. The system is required to store usage data for more axes than just requests, often for a long period of time—for compliance or customer service among others—and must offer acceptable read latency for any reasonable time slice of this data. For the purposes of this example, we are describing usage record capture and aggregation, not billing or business analytics.

Given these requirements, a reasonable architectural approach would be to perform local grouping of usage records on each API server receiving customer requests, and then on time/size boundaries before sending those off to a staging system that is write-scalable and, eventually, consistent. Once records are stored there, high durability is expected, so a dynamo-flavored key/value system would be a good choice for this tier. We can periodically run aggregation batch jobs on all this data. A MapReduce system or similar distributed batch-processing system makes sense as a starting point. Data must be retained for a long period of time—up to years—so using the batch-processing system provides an opportunity to aggregate data and store raw records along customer boundaries for easier retrieval later. This data can then be heavily compressed and sent to long-term, slow storage such as (at best) low-cost redundant disk drives or tape backups.

Alternative schemes could be considered, given slightly different requirements. For example, if having real-time visibility into usage data is important, then it would be reasonable instead to use a publish/subscribe system to route batches, or even every unique usage record, to some fleet of partitioned aggregation systems that keep "live" counts. The trade-off is that such a system is more costly to scale over time as it needs to be scaled nearly linearly with API requests.

Here are a few key takeaways on architecture:

- Decompose the "business" application into discrete services on the boundaries of fault domains, scaling, and data workload (r/w).
- Make as many things as possible stateless.
- When dealing with state, deeply understand CAP, latency, throughput, and durability requirements.

The remainder of this article briefly addresses the rest of the life cycle of such a system.

Implementation

Now that we have walked through the initial architecture of an image-processing service—which should be viable as an initial attempt—let's touch on a few misconceptions in implementing such a system. No opinion will be offered on operating systems, languages, frameworks, and others—simply a short set of guiding principles that may be useful.

When a new engineer sets out to implement such a system as described here, the typical first approach is immediately to survey available software—usually open source—and "glue" it together. Hopefully after taking the whirlwind tour of architecting such a system, you will recognize this approach as unwise. While it may work at development scale, every system is implemented with a set of assumptions by the engineer(s) who wrote it. These assumptions may or may not align with the requirements of your distributed system, but the point is, it is your responsibility to understand the assumptions that a technology makes, and its trade-offs, and then assess whether this software works in your environment. The common case is that some software will align with your assumptions, and some will not.

For the situations where you need to develop your own software to solve your system's problems, the next step should be assessing prior art in the space. Solutions can usually be found in some existing technology—often from longer ago than one would expect—but the assumptions of the system that implements this solution are different from yours. In such cases, break the system's strong points down into first principles, and "rebuild" on those. For example, in the authentication system described here, an existing Lightweight Directory Access Protocol (LDAP) solution would likely not be the right choice, given the assumptions of the protocol and existing LDAP system replication models, but we could certainly reuse many of the ideas of information management and authentication mechanisms, and implement those basic primitives in a way that scales for our environment.

As another example, if we need to implement or use a storage solution, many off-the-shelf solutions to this day involve some type of distributed file system (DFS) or storage area network (SAN), whether in the form of NFS (file), iSCSI (block), or some other protocol. But as described earlier, these systems all choose C in the CAP theorem, and in addition are often unable to scale past the capacity of a single node. Instead, the basic primitives necessary for many applications are present in modern object stores, but with different trade-offs (for example, no partial updates) that allow these storage systems to handle distributed scaling of some applications better.

Whether you are able to leverage existing components or build your own, all of them will need to consider how to manage production deployments at scale. I will not delve into the details here but point out only that configuration-management solutions should be looked at, and a system that can effectively manage deployments/rollback and state tracking of many systems is needed.

Testing and validation of a distributed system is extremely difficult—often as or more difficult than creating the system in the first place. This is because a distributed system simply has so many variables it is nearly impossible to control them all in isolation; therefore, for the most part, a system must be empirically validated rather than being proved theoretically correct. In particular, total outages in networking are relatively uncommon, but degraded service for brief periods of time is extremely likely. Such degradation often causes software to act in different ways from those encountered during development, resulting in the system software being the most likely cause of failure. Thus, the number-one cause of failures in distributed systems is without doubt heisenbugs—transient bugs that are often repeatable but diificult to reproduce, and dependent on some sequence of events in the system taking place.[7]

Validation of a distributed system involves many aspects, and certainly enough load needs to be driven through the system to understand where it "tips over" such that intelligent capacity planning and monitoring can be done. Even more so, though, failures and degraded states must be introduced regularly; failures can and will happen, and every component's dependencies must be taken away from it to understand the system acts as appropriate (even if that means returning errors to the customer) and the system recovers when the dependencies recover. The Netflix Chaos Monkey[2] is one of the best public references to such a methodology; automated failure/degradation to its test and production systems is considered essential to running its business.

Operations. Last, but certainly not least, are the operations of a distributed system. In order to run a system as described in this article, we must be able to capture and analyze all aspects of the hardware and operating systems, network(s), and application software—in short, as the mantra goes, "if it moves, measure it." This has two main aspects: realtime monitoring to respond to failures; and analytics—both realtime and historical—allowing operators, engineers, and analysts to understand the system and trends.

As may be obvious by this point, the amount of data a large distributed system can generate for operational data may, and likely will, exceed the amount of data that actually serves the business function. Storage and processing of this scale of data will likely require its own distributed system that can "keep up," which essentially brings us back to the beginning of this article: you'll need a distributed system to operate and manage your distributed system.

PaaS and Application Outsourcing

A final consideration is the recent adoption of Platform as a Service (PaaS), such as Google App Engine[6] or Heroku.[8] Such offerings have grown very popular as they reduce the engineering effort of delivering a vertical application to market, such as a traditional MVC (model-view-controller) Web application. By now, it should be obvious there are no magic bullets, and it is imprudent simply to hand over your system to a third party without due diligence. PaaS systems will indeed work for many applications—quite well, in fact—but again, only if your system requirements align with the feature set and trade-offs made by the PaaS provider; the former is fairly easy to evaluate, and the latter may be difficult or impossible. As an example, reference the recent public disclosure by Rap Genius[9] regarding Heroku's routing trade-offs. Ignoring all legal aspects, the problem encountered was ultimately a severe degradation of application latency caused by implementation trade-offs that suited the platform's requirements rather than the tenant's. It may well be the case the business was still correct in choosing PaaS, but the point is this issue was incredibly difficult to find a root cause, and the business hit it at the worst possible time: under production load as the application scaled up.

Conclusion

Distributed systems are notoriously difficult to implement and operate. Nonetheless, more engineers are finding themselves creating distributed applications as they move into running modern-day Web or mobile applications, or simply move an existing enterprise application onto a cloud service provider.

Without practical experience working on successful—and failed—systems, most engineers take a "hopefully it works" approach and attempt to string together off-the-shelf software, whether open source or commercial, and often are unsuccessful at building a resilient, performant system. In reality, building a distributed system requires a methodical approach to requirements along the boundaries of failure domains, latency, throughput, durability, consistency, and desired SLAs for the business application at all aspects of the application.

Related articles on queue.acm.org

Distributed Computing Economics
Jim Gray
http://queue.acm.org/detail.cfm?id=1394131

Monitoring and Control of Large Systems with MonALISA
Iosif Legrand, Ramiro Voicu, Catalin Cirstoiu, Costin Grigoras, Latchezar Betev, Alexandru Costan
http://queue.acm.org/detail.cfm?id=1577839

Condos and Clouds
Pat Helland
http://queue.acm.org/detail.cfm?id=2398392

References

1. Allspaw, J. *The Art of Capacity Planning.* O'Reilly Media, 2008; http://shop.oreilly.com/product/9780596518585.do.
2. Bennett, C. and Tseitlin, A. Netflix: Chaos Monkey released into the wild. Netflix Tech Blog; http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html.
3. Chandra, T., Griesemer, R. and Redstone, J. 2007. Paxos made live. In *Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing* (2007), 398-407; http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/paxos_made_live.pdf.
4. Dean, J. and Ghemawat, S. MapReduce: simplified data processing on large clusters. In *Proceedings of 6th Symposium on Operating Systems Design and Implementation I* (2004); http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/archive/mapreduce-osdi04.pdf.
5. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P, and Vogels, W. Dynamo: Amazon's highly available key-value store. In *Proceedings of the 21st ACM Symposium on Operating System Principles* (2007); http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf.
6. Google App Engine. Google Developers; https://developers.google.com/appengine/.
7. Gray, J. Why do computers stop and what can be done about it. Tandem Technical Report 85.7, 1985; http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.6561.
8. Heroku Cloud Application Platform; http://www.heroku.com/.
9. Somers, J. Heroku's ugly secret; http://rapgenius.com/James-somers-herokus-ugly-secret-lyrics.
10. Tarreau, W. Make applications scalable with load balancing; http://1wt.eu/articles/2006_lb/.

Mark Cavage is a software engineer at Joyent, where he works primarily on distributed-systems software and maintains several open source toolkits, such as ldapjs and restify. He was previously a senior software engineer with Amazon Web Services, where he led the Identity and Access Management team.

ACM ANNOUNCES NEW CHANGES TO EXPAND ACCESS TO PUBLICATIONS

ACM is pleased to announce important changes to our publications policy aimed at increasing exposure to and the impact of our publications on the global computing community. These changes further empower ACM authors to more widely distribute and make available their work, while at the same time taking advantage of the numerous benefits and prestige of publishing with ACM.

Some highlights of the new policy:

- ACM authors now have the option of retaining copyright and other important ownership rights by selecting one of ACM's "*Copyright Transfer*", "*Exclusive License to Publish*", or "*Non-Exclusive Permission to Publish*" Agreements
- ACM authors now have the option of making their work perpetually free to the world via the ACM DL platform by participating in the new ACM Open Access publication program.
- ACM Special Interest Groups (SIGs) now have the option of making their ACM conference proceedings freely available to the world on a limited basis via the ACM Digital Library, SIG, or conference websites.

For detailed information about these new options, as well as information about ACM's complete publications policy, please visit http://authors.acm.org.

acm Author Rights

CHOOSE your rights option
POST on your own websites
DISTRIBUTE via *ACM Author-Izer*
REUSE your own work
CREATE derivative works
RETAIN perpetual rights

Association for Computing Machinery

DOI:10.1145/2461256.2461275

Praise, pay, and promote crowd-member workers to elicit desired behavioral responses and performance levels.

BY OGNJEN SCEKIC, HONG-LINH TRUONG, AND SCHAHRAM DUSTDAR

Incentives and Rewarding in Social Computing

INCENTIVES AND REWARDS help align the interests of employees and organizations. They first appeared with the division of labor and have since followed the increasing complexity of human labor and organizations. As a single incentive measure always targets a specific behavior and sometimes additionally induces unwanted responses from workers, multiple incentives are usually combined to counteract the dysfunctional behavior and produce desired results. Numerous studies have shown the effectiveness[20] of different incentive mechanisms and their selective and motivational effects.[14] Their importance is reflected in the fact that most big and mid-size companies employ some kind of incentive measures.

Expansion of social computing[18] will include not only better exploitation of crowdsourcing[5] but also solutions that extend traditional business processes (see Figure 1); increasing research interest seems to confirm the trend.[3] Several frameworks aiming to support such new collaboration models are being developed (such as socially enhanced computing[6,7]). These new forms of social computing are intended to support greater task complexity, more intelligent task division, complex organizational and managerial structures for virtual teams, and virtual "careers." With envisioned changes, incentives will also gain importance and complexity to address workers' dysfunctional behavior. This new emphasis calls for automated ways of handling incentives and rewards. However, the social computing market is dominated by flat and short-lived organizational structures, employing a limited number of simple incentive mechanisms. That is why we view the state of the social computing market as an opportunity to add novel ways of handling incentives and rewards.

Here, we analyze incentive mechanisms and suggest how they can be used for next-generation social computing. We start with a classification of incentive mechanisms in the literature and in traditional business organizations, then identify elements that can be used as building blocks for any composite incentive mechanism and show the same elements are also used in social computing, even though the resulting schemes lack the complexity

» key insights

- **Existing social computing platforms lack the ability to formulate, compose, and automatically deploy appropriate incentive mechanisms needed for complex agent collaborations.**
- **Analyzing incentive mechanisms in traditional companies and in social computing platforms reveals how incentive mechanisms consist of simpler reusable elements that can be formally modeled.**
- **Formal modeling of incentive mechanisms allows composition, optimization, and deployment of portable and dynamically adaptable incentive schemes for social computing environments.**

ILLUSTRATION BY ALICIA KUBISTA/ANDRIJ BORYS ASSOCIATES

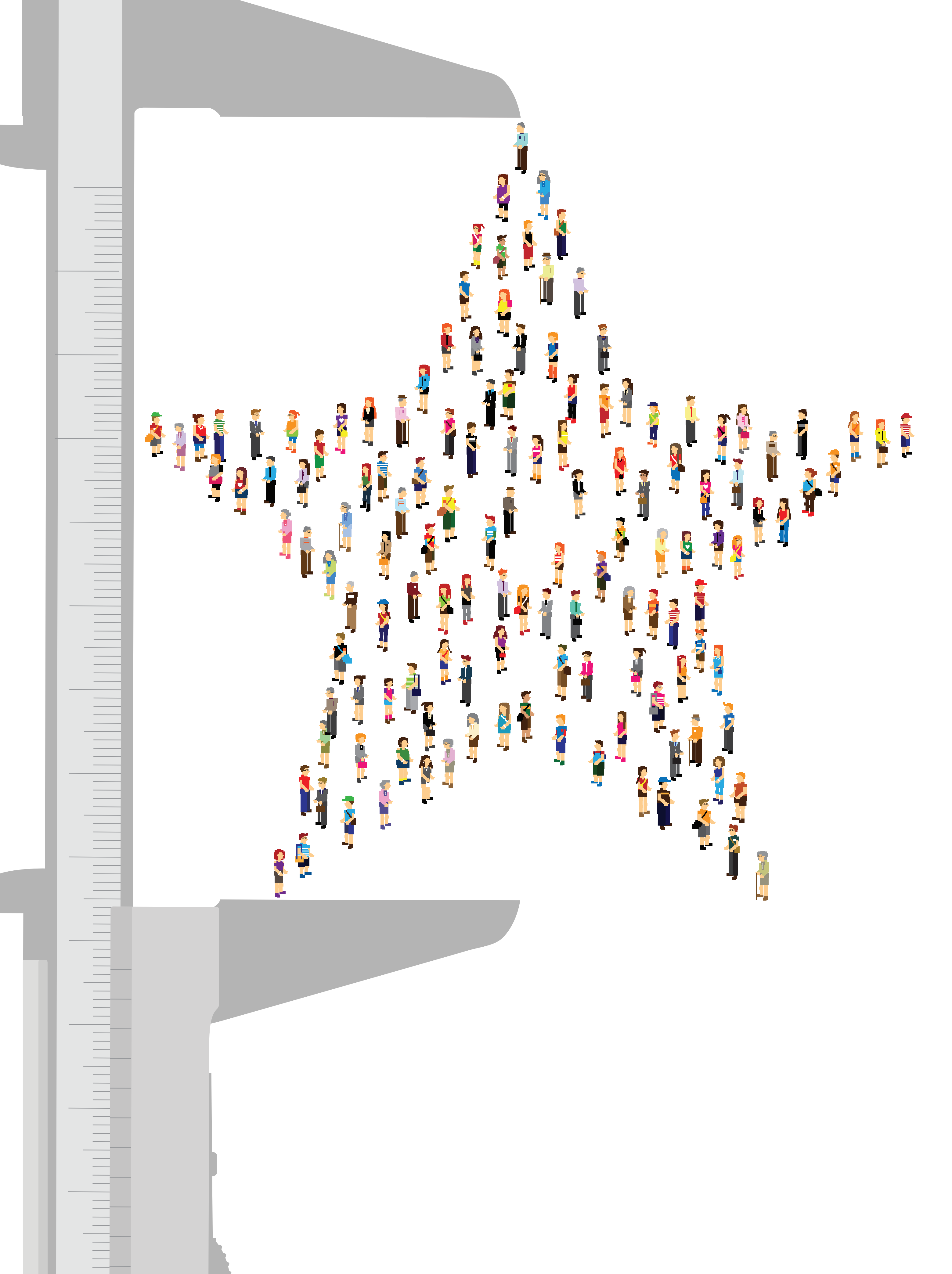
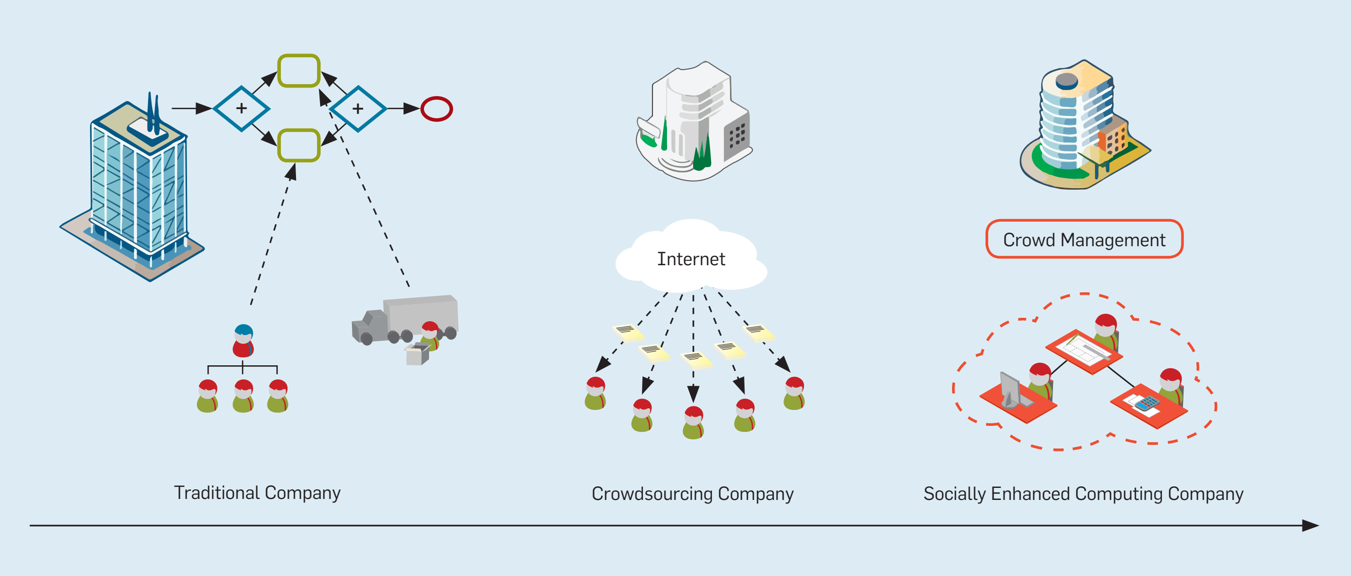


Figure 1. Social computing is evolving from social networks and crowdsourcing to include structured crowd organizations able to solve complex tasks.



needed to support advanced business processes; we conclude with our vision for future developments.

Related Work

In economics, incentives are predominantly investigated within the models set out in the Principal-Agent Theory,[13,20] introducing the role of a principal that corresponds to owners or managers who delegate tasks to a number of agents corresponding to employees (workers) under their supervision. The principal offers the agents an incentive to disclose part of their personal performance information (signal) to devise an appropriate contract.

Only a few articles in the computer science literature have addressed incentives and rewards, usually within specific application contexts (such as social networks and wiki systems,[9,32] peer-to-peer networks,[23] reputation systems,[22] and human micro-task platforms[16,17,28,29]). Much recent research aims to find suitable wage models for crowdsourcing.[11] However, to the best of our knowledge, the topic has not been comprehensively addressed.

Incentive Mechanisms

The incentive mechanisms we cover here involve most known classes of incentives used in different types of organizations: companies, not-for-profit (voluntary), engineering/design, and crowdsourcing. Different organizations employ different (combinations of) incentive mechanisms to stimulate specific responses from agents:

Pay per performance (PPP). The guiding principle says all agents are to be compensated proportionally to their contribution. Labor types where quantitative evaluation can be applied are particularly suitable. In practice, it shows significant, verifiable productivity improvements—25% to 40%—when targeting simple, repetitive production tasks, both in traditional companies[15] and in human intelligence tasks on Amazon's Mechanical Turk platform.[17] Other studies, as cited by Prendergast,[20] conclude that approximately 30% to 50% of productivity gains is due to the filtering and attraction of better workers, due to the selective effect of the incentive. This important finding explains why greater profit can be achieved even with relatively limited incentives. PPP is not suited for large, distributed, team-dependent tasks, where measuring individual contributions is inherently difficult. However, it is frequently used to complement other incentive mechanisms.

Quota systems and discretionary bonuses. With this mechanism, the principal sets a number of performance-metrics thresholds. When agents reach a threshold they are given a one-off bonus. Quota systems evaluate whether a performance signal surpasses a threshold at predefined points in time (such as annual bonuses). On the other hand, discretionary bonuses may be paid whenever an agent achieves a performance level for the first time (such as a preset number of customers).

Two phenomena[20] typically accompany this mechanism:

- The effort level always drops off following an evaluation if the agent views the time until the next evaluation as too long; and
- When the performance level is close to an award-winning quota, motivation is significantly greater.

Appropriate evaluation intervals and quotas must be set in such a way that they are achievable with a reasonable amount of additional effort, though not too easily. The two parameters are highly context-dependent, so can be determined only after observing historical records of employee behavior in a particular setup. Ideally, these parameters are dynamically adjustable.

Deferred compensation. This mechanism is similar to a quota system, in that an evaluation is made at predefined points in time. The subtle but important difference is that deferred compensation takes into account three points in time: t_0,t_1,t_2. At t_0 an agent is promised a reward for successfully passing a deferred evaluation at t_2. The evaluation takes into account the period of time $[t_1,t_2]$, not just the current state at t_2. In case $t_1 = t_0$ the evaluation covers the entire interval.

Deferred compensation is typically used for incentivizing agents working on complex, long-lasting tasks. The advantage is it allows more objective assessment of an agent's performance at a particular time. Agents are also given enough time $[t_0,t_1]$ to adapt to the new conditions, then prove the quality of their work over some period of time $[t_1,t_2]$. The disadvantage of this mechanism is it is not always applicable, since agents are not always able to wait for a significant portion of their compensation. A common example of this mechanism is the "referral bonus," or a reward for employees who recommend or attract new employees or partners to the company.

Relative evaluation. Although this mechanism can involve many variations the common principle is that an entity is evaluated with respect to other entities within a specified group. The entity can be a human, a movie, or a product. The relative evaluation is used mainly for two reasons:

- By restricting the evaluation to a closed group of individuals, it removes the need to set explicit, absolute performance targets in conditions where the targets are not easily set due to the dynamic and unpredictable nature of the environment; and
- It has been empirically proved that people respond positively to competition and comparison with others (such as in Tran et al.[30]).

Promotion. Empirical studies (such as Van Herpen et al.[31]) confirm that the prospect of a promotion increases motivation. A promotion is the result of competition for a limited number of predefined prizes. Promotion schemes are usually treated under the tournament theory,[14] though there are other models, too. The prize is a better position in an organization's hierarchy, bringing higher pay, more decision-making power, and greater respect and esteem. Promotions include basic

Table 1. Adoption of incentive mechanisms in different business environments: + = low, ++ = medium, +++ = high; application considerations (right).

	Usage Environments			Application Considerations			
	Traditional Company						
Mechanism	**SME**	**Large Enterprise**	**Social Computing**	**Positive Application Conditions**	**Negative Application Conditions**	**Advantages**	**Disadvantages**
Pay Per Performance	++	+++	+++	quantitative evaluation possible	large, distributed, team-dependent tasks; measurement inaccuracy; when favoring quality over quantity	fairness; effort continuity	oversimplification; decreased solidarity among workers
Quota/ Discretionary Bonus	+	+++	+	recurrent evaluation intervals	constant level of effort needed	allows peaks/ intervals of increased performance	effort drops after evaluation
Deferred Compensation	+	+++	+	complex, risky, long-lasting tasks	subjective evaluation; short consideration interval	better assessment of achievements; paying only after successful completion	workers must accept risk and wait for compensation
Relative Evaluation	+	++	+++	cheap group-evaluation method available	subjective evaluation	no absolute performance targets; eliminates subjectivity	decreases solidarity; can discourage beginners
Promotion	++	+++	+	need to elicit loyalty and sustained effort; when subjective evaluation is unavoidable	flat hierarchical structure	forces positive selection; eliminates centrality bias	decreases solidarity
Team-based Compensation	+	++	+	complex, cooperative tasks; inability to measure individual contributions	when retaining the best individuals is priority	increases cooperation and solidarity	disfavors best individuals
Psychological	+	+	++	stimulate competition; stimulate personal satisfaction	when cooperation must be favored	cheap implementation	limited effect on best and worst workers (anchoring effect)

ideas from relative evaluation and quota systems. They eliminate centrality bias and enforce positive selection. The drawback is that by valuing individual success, agents can be de-motivated from helping each other and engaging in collaborations. They often incorporate subjective evaluation methods, though other evaluation methods are also possible in rare instances.

Team-based compensation. This mechanism is used when the contributions of individual agents in a team environment are not easily identified. With it, the entire team is evaluated and rewarded, with the reward split among team members. The reward can be split equally or by differentiating individual efforts within the team. The latter is a hybrid approach combining a team-based incentive, together with an incentive mechanism targeted at individuals, to eliminate dysfunctional behavior. Some studies (such as Pearsall et al.[19]) show this approach is indeed more effective than pure team-based compensation. One way to avoid having to decide on the amount of compensation is to tie it to the principal's profit, and is called "profit sharing." Team-based compensation is also susceptible to different dysfunctional behavioral responses. Underperforming agents effectively hide within the group, while the performance of the better-performing agents is diluted. Moreover, teams often exhibit the free-rider phenomenon,[12] where individuals waste more resources (such as time, materials, and equipment) than they would if individual expenses were measured. Minimizing these negative effects is the primary challenge when applying this mechanism.

Psychological incentive mechanisms. Psychological incentives are the most elusive, making them difficult to define and classify, since they often complement other mechanisms and can be described only in terms of psychological actions. A psychological incentive must relate to human emotions and be advertised by the principal and be perceived by the agent. The agent's perception of the incentive affects its effectiveness. As this perception is context-dependent, choosing an adequate way of presenting the incentive is not trivial; for example, choosing and promoting an "employee of the month" is effective in societies where the sense of common good is highly valued. In more individually oriented environments competition drives performance. A principal may choose to exploit this fact by sharing comparisons with the agents. Acting on human fear is a tactic commonly (mis)used (such as through the threat of dismissal or downgrading). Psychological incentives have long been used in video games, as well as in more serious games, to elicit player dedication and motivation. Such techniques (including gamification[4]) are also used to make boring tasks (such as product reviews and customer feedback) feel more interesting and appealing (see Table 1).

The effort level always drops following an evaluation if the agent views the time until the next evaluation as too long.

Analyzing Incentive Mechanisms

No previous work has analyzed incentives past the granularity of incentive mechanisms, preventing (development of) generic handling of incentives in information systems. Our goal is to identify finer-grain elements that can be modeled individually and used in information systems to compose and encode the described incentive mechanisms (see Figure 2). Such a conceptual model would allow specification, execution, monitoring, and adaptation of various rewarding and incentive mechanisms for virtual teams of humans.

Each incentive mechanism described earlier can be modeled using three incentive elements:

Evaluation method. Provides input on agent performance to be evaluated in the logical context defined in the incentive condition;

Incentive condition. Contains the business logic for certain rewarding actions; and

Rewarding action. Is meant to influence future behavior of agents.

Though we describe these elements informally here, their true power lies in the possibility of being formally modeled. An evaluation method can ultimately be abstracted to an evaluation function, incentive condition to a logical formula, and rewarding action to a function, structural transformation, or external event. These abstractions allow us to formally encode each incentive mechanism and thus program many real-world reward strategies for crowds of agents working on tasks ranging from simple image tagging to

Table 2. Application and composability considerations for evaluation methods.

		Application Considerations			Composability			
Evaluation Methods		Advantages	Disadvantages	Active Human Participation	Issues	Alleviated By	Solving	Typical Use
Individual	Quantitative	fairness, simplicity, low cost	measurement inaccuracy	no	multitasking	peer evaluation; indirect evaluation; subjective evaluation	issues due to subjectivity	pay per performance; quota systems; promotion; deferred compensation
	Subjective	simplicity, low cost	subjectivity; inability to assess different aspects of contribution	yes	centrality bias; leniency bias; deliberate low-scoring; embellishment; rent-seeking activities	incentivizing decision maker to make honest decisions (such as through peer evaluation)	multitasking	relative evaluation; promotion
	Peer	fairness; low cost in social computing environment	active participation required	yes	preferential attachment; coordinated dysfunctional behavior of voters	incentivizing peers (such as also by peer evaluation)	multitasking; issues due to subjectivity	relative evaluation; team-based compensation; psychological
Group	Indirect	accounts for complex relations among agents and their artifacts	evaluation-algorithm cost of development and maintenance	no	depends on algorithm used; fitting data to the algorithm	peer voting; better implementation of algorithm	issues due to subjectivity; peer-evaluation issues	relative evaluation; psychological; pay per performance

modular software development.

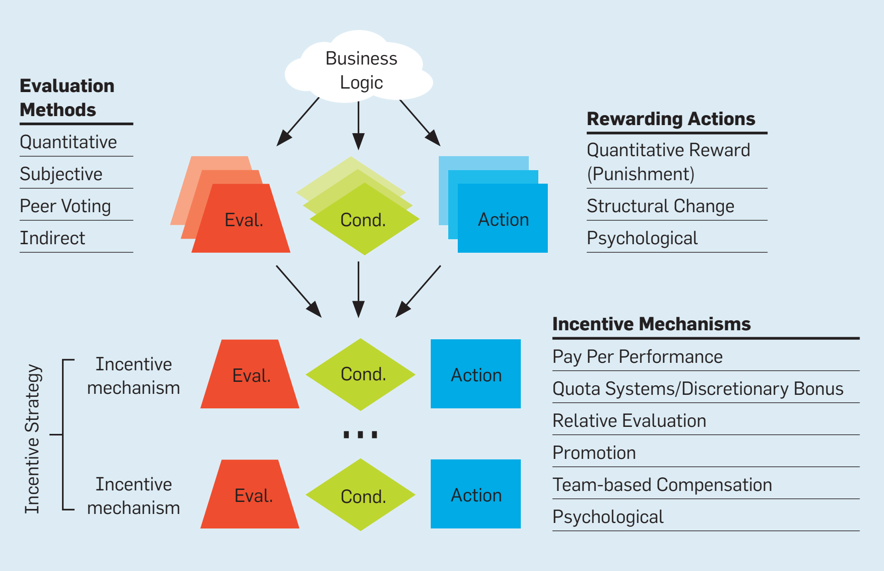
Individual Evaluation Methods

Quantitative evaluation. Quantitative evaluation represents the rating of individuals based on the measurable properties of their contribution. Some labor types are suitable for precisely measuring an agent's individual contributions, in which case the agent can be evaluated on number of units processed, but apart from the most primitive types of labor, evaluating an agent's performance requires evaluating different aspects of performance, or measurable signals, the most common being productivity, effort, and product quality. Different measures are usually taken into consideration with different weights, depending on their importance and measurement accuracy.

Quantitative evaluation is attractive because it does not require human participation and can be implemented entirely in software. Associated problems are measurement inaccuracy and difficulty choosing proper signals and weights. An additional problem is a phenomenon called multitasking, which, in spite of its counterintuitive name, refers to agents putting most of their effort into tasks subject to incentives while neglecting other tasks, subsequently damaging overall performance.[10]

Subjective evaluation. When important aspects of human work are understandable and valuable to humans only, we need to substitute an objective measurement with a human (subjective) assessment of work quality. In this case a human acts as a mapping function that quantifies human-oriented work by combining all undefinable signals into one subjective assessment signal. Even though subjective evaluation is implemented simply and cheaply, it is also inherently imprecise and prone to dysfunctional behavioral responses. Phenomena observed in practice[20] include:

Figure 2. Incentive strategies consist of smaller, easily modeled components.



Centrality bias. Ratings concentrated around some average value, so not enough differentiating of "good" and "bad" workers;

Leniency bias. Discomfort rating "bad" workers with low marks; and

Rent-seeking activities. Actions taken by employees with the goal of increasing the chances of getting a better rating from a manager, often including personal favors or unethical behavior.

Group Evaluation Methods

Peer evaluation (peer voting). Peer evaluation is an expression of collective intelligence where members of a group evaluate the quality of other members. In the ideal case, the aggregated, subjective scores represent a fair, objective assessment. The method alleviates centrality and leniency bias since votes are better distributed, the aggregated scores cannot be subjectively influenced, and activities targeting a single voter's interests are eliminated. Engaging a large number of professional peers to evaluate different aspects of performance leaves fewer options for multitasking.

This method also suffers from a number of weaknesses; for example, in small interconnected groups voters may be unjust or lenient for personal reasons. They may also feel uncomfortable and exhibit dysfunctional behavior if the person being judged knows their identity. Therefore, anonymity is often a favorable quality. Another way of fighting dysfunctional behavior is to make voters subject to incentives; votes are compared, and those that stand out discarded. At the same time, each agent's voting history is monitored to prevent consistent unfair voting.

When the community consists of a relatively small group of voted persons and a considerably larger group of voters and both groups are stable over time, this method is particularly favorable. In such cases, voters have a good overview of much of the voted group. Since the relationship voter-to-voted is unidirectional and probably stable over time, voters do not have an interest in exhibiting dysfunctional behavior, a pattern common on the Internet today.

The method works as long as the size of the voted group is small. As the voted group increases, voters are unable to acquire all the new facts needed to pass fair judgment. They then opt to rate better those persons or artifacts they know or feel have good reputations (see Price[21]), a phenomenon known as "preferential attachment," or colloquially "the rich get richer." It can be seen on news sites that attract large numbers of user comments. Newly arriving readers usually tend to read and vote only the most popular comments, leaving many interesting comments unvoted.

In non-Internet-based businesses, cost is the major obstacle to applying this method, in terms of both time and money. Moreover, it is technically challenging, if not impossible, to apply it often enough and with appropriate voting groups. However, the use of information systems, the Internet, and social networks now makes possible a drastic decrease in application costs. A number of implementations exist on the Internet (such as Facebook's Like button, binary voting, star voting, and polls), but lacking is a unified model able to express their different flavors and specify the voters and voted groups.

Indirect evaluation. Since human performance is often difficult to define and measure, evaluating humans is commonly based on properties and relations among the artifacts they produce. As the artifacts are always produced for consumption by others, determining quality is ultimately left to the community. Artifacts are connected through various relations (such as contains, refers-to, and subclass-of) among themselves, as well as with users (such as author, owner, and consumer). The method of mapping properties and relations of artifacts to scores is nontrivial. An algorithm (such as Google's PageRank) tracks relations and past interactions of agents or their artifacts with the artifact being evaluated and calculates the score. A tailor-made algorithm must usually be developed or an existing one adapted to fit a particular environment. The major difference from peer evaluation is the agent does not actively evaluate the artifact, and hence the algorithm is not dependent on interacting with the agent.

The method's advantages and drawbacks fully depend on the properties of the applied algorithm. If the algorithm is suitable it exhibits fairness and prevents false results. The cost of the method depends in turn on the cost of developing, implementing, and running the algorithm. A common problem involves users who know how the algorithm works, then try to deceive it by outputting dummy artifacts with the sole purpose of increasing their scores. Detecting and preventing such attempts requires amending the algorithm, further increasing costs; Table 2 lists common application and composability considerations for these evaluation methods, as well as how drawbacks of a particular evaluation method can be alleviated by combining it with other methods.

Table 3. Incentive mechanisms used by social computing companies.

Incentive Strategy	No. of Companies	Percentage
Relative Evaluation	75	54%
Pay Per Performance	46	33%
Psychological	23	16%
Quota/Discretionary Bonus	12	9%
Deferred Compensation	10	7%
Promotion	9	6%
Team-based Compensation	3	2%

Table 4. Number of incentive mechanisms used by social computing companies; a majority of surveyed companies and organizations employ only one mechanism.

No. of Mechanisms	No. of Companies	Percentage
1	116	83%
2	15	11%
3	6	4%
≥4	3	2%

Rewarding Actions

Agents' future behavior can be influenced through rewarding actions:

Rewards. Rewards can be modeled as quantitative changes in parameters associated with an agent; for example, a parameter can be the wage amount, which can be incremented by a bonus or decreased by a penalty;

Structural changes. Structural changes are an empirically proven[31] motivator. A structural change does not strictly imply position advancement/downgrading in traditional tree-like management structures but does include belonging to different teams at different times or collaborating with different people; for example, working on a team with a distinguished individual can diversify an agent's experience and boost the agent's career. One way to model structural changes is through graph rewriting;[2] and

Psychological actions. Though all incentive actions have a psychological effect, psychological actions are only those in which an agent is influenced solely by information; for example, we may decide to show agents only the results of a couple of better-ranking agents rather than the full rankings. This way, the agents will not know their position in the rankings and can be beneficial in two ways: prevent the anchoring effect[17] for agents in the top portion of the rankings and prevent discouragement of agents in the lower portion. Psychological actions do not include explicit parameter or position change, but the diversity of presentation options means defining a unified model for describing different psychological actions is an open challenge. Effects of these actions are difficult to measure precisely, but apart from empirical evidence (such as Frey and Jegen[8]), their broad adoption on the Internet today is a clear indication of their effectiveness.

Incentive Conditions

Incentive conditions state precisely how, when, and where to apply rewarding actions, with each action consisting of at most three components, or subconditions:

Parameter. Expresses a subcondition in the form of a logical formula over a specified number of parameters describing an agent;

Since human performance is often difficult to define and measure, evaluating humans is commonly based on properties and relations among the artifacts they produce.

Time. Helps formulate a condition over an agent's past behavior; and

Structure. Filters out agents based on their relationships and can be used to select members of a team or all collaborators of a particular agent.

Using these components at the same time helps make it possible to specify a complex condition (such as "target the subordinates of a specific manager, who over the past year achieved a score higher than 60% in at least 10 months").

Incentive conditions are part of the business logic, and as such are stipulated by HR managers. However, a small company can take advantage of good practices and employ pre-made incentive models (patterns) adapted to its needs. Feedback information obtained through monitoring execution of rewarding actions can help adapt condition parameters.

In Real-World Social Computing Platforms

In the first half of 2012 we surveyed more than 1,600 Internet companies and organizations worldwide that described themselves through keywords like "social computing" and "crowdsourcing," providing a solid overview of the overall domain of social computing. However, we were interested only in those employing incentive measures. Therefore, we manually investigated their reward/incentive practices (such as types of awards, evaluation methods, rules, and conditions) as stated on company websites, classifying them according to the previously described classifications. Overall, we identified and examined 140 companies and organizations using incentive measures.

Survey results. We found it striking that 59 of the 140 companies (42%) used a simple "contest" business model employing a relative evaluation incentive mechanism in which a creative task is deployed to the crowd. Each crowd member (or entity) then submits a design. The best design in the vast majority of cases is chosen through subjective evaluation (85%). That was expected, since the company buying the design reserves the right to decide the best design. In fact, in many cases, it was the only possible choice. When using peer evaluation, a company delegates the decision as to the best design to the crowd of peers while taking the risk of producing and selling the design. In some cases (such as a programming contest), the artifacts are evaluated quantitatively through automated testing procedures. Worth noting is that peer or quantitative evaluation produces quantifiable user ratings. In such cases, individuals are better motivated to take part in future contests, even if they feel they cannot win, because they can use their ranking as a personal quality proof when applying for other jobs or even as personal prestige.

Apart from the 59 organizations running contests, relative evaluation is used by another 16 organizations, usually combined with other mechanisms. This makes relative evaluation by far the most widely used incentive mechanism in social computing today (54% of those we surveyed) (see Table 3). This is in contrast with its use in traditional (non-Internet-based) businesses, where it is used considerably less,[1] as implementation costs are much greater.

The other significant group includes companies that pay agents for completing human micro-tasks. We surveyed 46 such companies (33%). Some are general platforms for submitting and managing any kind of human-doable tasks (such as Amazon's Mechanical Turk). Others offer specialized human services, most commonly writing reviews, locating software bugs, translating, and performing simple, location-based tasks. What all these companies have in common is the PPP mechanism. Quantitative evaluation is the method of choice in most cases (65%) in this group.

Quantitative evaluation sometimes produces binary output (such as when submitting successful/unsuccessful steps to reproduce a bug). The binary output allows expressing two levels of the quality of work, so agents are rewarded on a per-task basis for each successful completion. In this case, the company usually requires no entry tests for joining the contributing crowd. In other cases, establishing work quality is not easy, and the output is proportional to the quantity of finer-grain units performed (such as word count in translation tasks), though agents are usually asked to complete entry tests; the pay rate for subsequent work is determined by the test results. Other evaluation methods include subjective and peer/indirect evaluation, both at 17%. Interesting to note is how rarely peer evaluation is employed for double-checking results, as companies find it cheaper to test contributors once, then trust their skills later on. However, as companies start to offer more complex human tasks, quality assurance becomes imperative, so we expect so see a rise in peer and indirect evaluation.

Only three companies combined four or five different mechanisms (see Table 4). The most well known is uTest.com; with a business model requiring a large crowd of dedicated professionals, it is clear why it employs more than just simple PPP.

ScalableWorkforce.com is the only company we studied that advertises the importance of crowd (work-force) management, offering tools for crowd management on Amazon's Mechanical Turk to its clients. The tools allow for tighter agent collaboration (fostering a sense of community among workers), workflow management, performance management, and elementary career building.

Of the 140 organizations we surveyed, 12 (8.5%) rely uniquely on psychological mechanisms to assemble and improve their agent communities. Their common trait is their reliance on the indirect influence of rankings in an agent's (non-virtual) professional life; for example, avvo.com attracts large numbers of lawyers in the U.S. who offer a free response and advice to people visiting the website. Quality and timeliness of professionals' responses affect their reputation rankings on avvo.com, which can be used as an advertisement to attract paying customers to their private practices. Another interesting example involves companies like crowdpark.de and prediculous.com that ask their users to "predict" the future by betting on upcoming events with virtual currency. Users with the best predictions over time earn virtual trophies (badges), the only incentive for participation. Crowdsourced odds are also useful for adjusting odds in conventional betting involving real money.

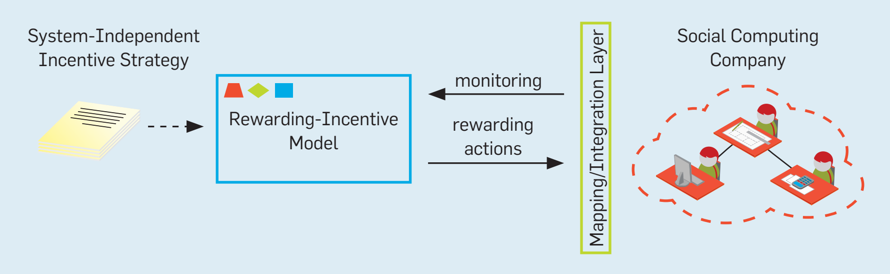
Table 5. Evaluation methods, excluding companies running creative contests.

Evaluation Method	No. of Companies	Percentage
Quantitative Evaluation	51	63%
Peer Voting + Indirect	35	43%
Subjective Evaluation	14	17%

Table 6. Companies using different evaluation methods (columns) within different incentive mechanisms (rows) as of early 2012; they may not be the primary mechanisms used by these companies.

	Quantitative	Subjective	Peer	Indirect
Pay Per Performance	mturk.com	content.de	crowdflower.com	translationcloud.net
Quota/Discretionary Bonus	gild.com		carnetdemode.fr	
Deferred Compensation	advisemejobs.com	bluepatent.com	crowdcast.com	
Relative Evaluation	netflixprize.com	designcrowd.com	threadless.com	topcoder.com
Promotion	utest.com	scalableworkforce.com	kibin.com	
Psychological Incentives	crowdpark.de	battleofconcepts.nl	avvo.com	
Team-based Compensation		mercmob.com	geniuscrowds.com	

Figure 3. Conceptual scheme of a system able to translate portable incentive strategies into concrete rewarding actions for different social computing platforms.



Team-based compensation was used by only three companies we surveyed; for example, mercmob.com encourages formation of virtual human teams for various tasks. Agents express confidence in the successful completion of a task by investing part of a limited number of their "contracts," or a type of local digital currency. When invested, the contracts are tied to the task, motivating the agents who accept the task to do their best to self-organize as a team and attract others to join the effort. If in the end the task is completed successfully each agent gets a monetary reward proportional to the number of invested contracts.

Discretionary bonuses or quota systems are used by 11 companies (8%). However, they are always used in combination with another mechanism, usually PPP (64%), as in traditional companies.

Deferred compensation is used by 7% of the companies, usually as their only incentive mechanism; for example, Bluepatent.com crowdsources the tasks of locating prior art for potential patent submissions. The agents (researchers) are asked to find and submit relevant documents proving the existence of prior art. Deciding on the validity and usefulness of such documents is an intricate task, hence the decision on compensation is delayed until an expert committee decides on it. Advisemejobs.com pays out classical referral bonuses to agents suggesting appropriate job candidates.

Only 7% of our surveyed companies offer career advancement combined with other incentive mechanisms. As the crowd structure is usually plain, career advancement usually means higher status, implying a higher wage. We encountered only two cases where advancement also meant structural change, with an agent taking responsibility for leading or supervising lower-ranked agents.

In traditional companies deciding on a particular employee's promotion is usually a matter of subjective evaluation by the employee's superiors. With the promotion being the most commonly employed traditional incentive, the subjective evaluation is also the most commonly used evaluation method. However, if we remove the companies running creative contests, where the artistic nature of the artifacts forces use of subjective evaluation, we see a reversal of the trend in social computing. Subjective evaluation trails quantitative and peer evaluation (see Table 5), as explained by the fact that information systems enable cheaper measurement of different inputs and setting up peer-voting mechanisms.

Only a small number of the companies and organizations we surveyed employ a combination of incentive mechanisms. Locationary.com uses agents around the world to expand and maintain a global business directory by adding local business information, employing two basic incentive mechanisms, aided by a number of supporting ones: The first is the so-called conditional PPP; with every new place added and/or corrected, agents win "lottery tickets" that increase the chances of winning a reward in a lottery, though a minimum number of tickets is needed to enter the draw. The second is team-based compensation. Locationary.com shares 50% of the revenues obtained from each directory entry with its agents. Any agent adding new information about a business obtains a number of shares of that directory entry. The reward is then split among agents proportionally to the number of shares they possess. Additionally, each entry in the directory must be approved through votes by trusted agents. Each agent has a trust score calculated by indirect evaluation that accounts for numerous factors like trust scores of voters, number of approved and rejected entries, and freshness of data. Trust influences the number of tickets awarded, thus affecting the odds of winning an award; the actual payout is limited to the agents with a certain level of trust.

Locationary.com uses a combination of PPP and a quota system to motivate overall agent activity. Team-based compensation is used to incentivize adding high-end clients to the directory first. If an agent is first to add detailed information about, say, a hotel on the Mediterranean Sea, then, in addition to lottery tickets, that agent can expect appreciable income from the hotel's advertising revenue. Adding a local fast-food restaurant could bring the same number of lottery tickets but probably no advertising revenue. Peer voting serves to maintain data accuracy and quality, while indirect evaluation (expressed through trust) identifies and keeps high-quality contributors. In the end, we also see an example of deferred compensation, with money paid to contributors after some length of time but only if at the moment of payout they still have a satisfactory trust level. This example demonstrates how different mechanisms are used to target different agent behaviors and how to compose them to achieve their full effect; Table 6 outlines several companies employing different evaluation methods within a number of incentive mechanisms.

Conclusion

With creativity contests and microtask platforms dominating the social computing landscape the organizational structure of agents is usually flat or very simple; hierarchies, teams of agents, and structured agent collaborations are rare. In such environments, most social computing companies need to use only one or two simple incentive mechanisms, as in Table 4. Promotion, commonly used in traditional companies, is rarely found in social computing companies. The reason is the short-lived nature of transactions between agents and the social computing companies. For the same reason, team-based compensation is also poorly represented. The idea of building a "career in the cloud" is a distant dream. On the other hand, most traditional companies combine elaborate mechanisms to elicit particular responses from their agents and retain quality employees.[20] The mechanisms complement one another to cancel out individual drawbacks.

Our survey shows that as the cost of quantitative, peer, and indirect evaluation has decreased, relative evaluation and PPP have become the most popular incentive mechanisms among social computing companies. Subjective evaluation, though well represented, is found largely within companies that base their business models on organizing creativity contests and that psychological incentives and gamification approaches are gaining ground. We expect them to achieve their full potential as amplifiers for other incentive mechanisms.

The expected growth in complexity of business processes and organizational structures due to social computing will require novel automated ways of handling the behavior of agent crowds. That is why it is necessary to develop models of incentive mechanisms and frameworks fitting existing business models and real-world systems (such as workflow, human-provided services,[26] and crowdsourcing), while supporting composability and reusability of incentive mechanisms.

Such systems must be able to monitor crowds of agents and perform runtime adaptation of incentive mechanisms to prevent diverse negative effects (such as free riding, multitasking, biasing, anchoring, and preferential attachment), switching when needed between different evaluation methods, rewarding actions, and incentive conditions at runtime, while minimizing overall costs. This way, particular agent sub-groups can be targeted more efficiently.

Additional benefits include:

Optimization. Historical data can help detect performance bottlenecks, preferred team compositions, and optimal wages. Additionally, we can choose the optimal composition of incentive mechanisms, opening up novel possibilities for achieving indirect automated team adaptability through incentives;

Reusability. For certain business models, proven incentive patterns cut time and cost, with incentive patterns tweaked to fit particular needs based on feedback obtained through monitoring;

Portability. By generalizing and formally modeling incentive mechanisms, we can encode them in a system-independent manner; that way, they become usable on different underlying systems, without having to write more system-specific programming code (see Figure 3); and

Incentives as a service. Managing rewards and incentives can be offered remotely as a Web service.

We are developing an incentive framework supporting these functionalities[24,25] evaluated on social-compute-unit systems[27] for maintaining large-scale distributed cloud-software systems.

Acknowledgment

This work has been supported by the EU FP7-ICT 600854 SmartSociety Project and the Vienna PhD School of Informatics. C

References

1. Armstrong, M. *Armstrong's Handbook of Reward Management Practice: Improving Performance Through Reward.* Kogan Page Publishers, London, 2010.
2. Baresi, L. and Heckel, R. Tutorial introduction to graph transformation: A software engineering perspective. In *Proceedings of the First International Conference on Graph Transformation* (Barcelona). Springer-Verlag, London, 2002, 402–429.
3. Davis, J.G. From crowdsourcing to crowdservicing. *IEEE Internet Computing 15*, 3 (2011), 92–94.
4. Deterding, S., Sicart, M., Nacke, L., O'Hara, K., and Dixon, D. Gamification: Using game design elements in non-gaming contexts. In *Proceedings of the 2011 Annual Conference Extended Abstracts on Human Factors in Computing Systems* (Vancouver, B.C.). ACM Press, New York, 2011, 2425–2428.
5. Doan, A., Ramakrishnan, R., and Halevy, A.Y. Crowdsourcing systems on the World-Wide Web. *Commun. ACM 54*, 4 (Apr. 2011), 86–96.
6. Dustdar, S., Schall, D., Skopik, F., Juszczyk, L., and Psaier, H. (Eds.) *Socially Enhanced Services Computing: Modern Models and Algorithms for Distributed Systems.* Springer, 2011.
7. Dustdar, S. and Truong, H. L. Virtualizing software and humans for elastic processes in multiple clouds: A service-management perspective. *International Journal of Next-Generation Computing 3*, 2 (2012).
8. Frey, B.S. and Jegen, R. Motivation crowding theory. *Journal of Economic Surveys 15*, 5 (2001), 589–611.
9. Hoisl, B., Aigner, W., and Miksch, S. Social rewarding in wiki systems: Motivating the community. In *Proceedings of the Second International Conference on Online Communities and Social Computing* (Beijing). Springer, 2007, 362–371.
10. Holmstrom, B. and Milgrom, P. Multitask principal-agent analyses: Incentive contracts, asset ownership, and job design. *Journal of Law, Economics, and Organization 7* (1991), 24–52.
11. Horton, J.J. and Chilton, L.B. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM Conference on Electronic Commerce* (Cambridge, MA). ACM Press, New York, 2010, 209–218.
12. Kerrin, M. and Oliver, N. Collective and individual improvement activities: The role of reward systems. *Personnel Review 31*, 3 (2002), 320–337.
13. Laffont, J.-J. and Martimort, D. *The Theory of Incentives.* Princeton University Press, Princeton, NJ, 2002.
14. Lazear, E.P. Personnel economics: The economist's view of human resources. *Journal of Economic Perspectives 21*, 4 (2007), 91–114.
15. Lazear, E.P. Performance pay and productivity. *The American Economic Review 90*, 5 (2000), 1346–1361.
16. Little, G., Chilton, L.B., Goldman, M., and Miller, R.C. TurKit: Tools for iterative tasks on Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation* (Paris). ACM Press, New York, 2009, 29–30.
17. Mason, W. and Watts, D.J. Financial incentives and the performance of crowds. In *Proceedings of the ACM SIGKDD Workshop on Human Computation* (Paris). ACM Press, New York, 2009, 77–85.
18. Parameswaran, M. and Whinston, A.B. Social computing: An overview. *Communications of the Association for Information Systems 19* (2007), 762–780.
19. Pearsall, M.J., Christian, M.S., and Ellis, A.P.J. Motivating interdependent teams: Individual rewards, shared rewards, or something in between? *Journal of Applied Psychology 95*, 1 (2010), 183–191.
20. Prendergast, C. The provision of incentives in firms. *Journal of Economic Literature 37*, 1 (1999), 7–63.
21. Price, D.D.S. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American Society for Information Science 27*, 5 (1976), 292–306.
22. Resnick, P., Kuwabara, K., Zeckhauser, R., and Friedman, E. Reputation systems: Facilitating trust in Internet interactions. *Commun. ACM 43*, 12 (Dec. 2000), 45–48.
23. Sato, K., Hashimoto, R., Yoshino, M., Shinkuma, R., and Takahashi, T. Incentive mechanism considering variety of user cost in P2P content sharing. In *Proceedings of the IEEE Global Telecommunications Conference* (New Orleans). IEEE, 2008, 1–5.
24. Scekic, O., Truong H.L., and Dustdar, S. Modeling rewards and incentive mechanisms for social BPM. In *Proceedings of the 10th International Conference on Business Process Management* (Tallinn, Estonia, Sept.). Springer, 2012, 150–155.
25. Scekic, O., Truong, H.L., and Dustdar, S. Programming incentives in information systems. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering* (Valencia, Spain, June 2013).
26. Schall, D., Dustdar, S., and Blake, M.B. Programming human and software-based Web services. *IEEE Computer 43*, 7 (2010), 82–85.
27. Sengupta, B., Jain, A., Bhattacharya, K., Truong, H. L., and Dustdar, S. Who do you call? Problem resolution through social compute units. In *Proceedings of the 10th International Conference on Service-oriented Computing* (Shanghai). Springer, Berlin, Heidelberg, 2012, 48–62.
28. Shaw, A.D., Horton, J.J., and Chen, D.L. Designing incentives for inexpert human raters. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work* (Hangzhou, China). ACM Press, New York, 2011, 275–284.
29. Tokarchuk, O., Cuel, R., and Zamarian, M. Analyzing crowd labor and designing incentives for humans in the loop. *IEEE Internet Computing 16*, 5 (2012), 45–51.
30. Tran, A. and Zeckhauser, R. *Rank As An Incentive.* HKS Faculty Research Working Paper Series RWP09-019, John F. Kennedy School of Government, Harvard University, Cambridge, MA, 2009.
31. Van Herpen, M., Cools, K., and Van Praag, M. Wage structure and the incentive effects of promotions. *Kyklos 59*, 3 (2006), 441–459.
32. Yogo, K., Shinkuma, R., Konishi, T., Itaya, S., Doi, S., Yamada, K., and Takahashi, T. Incentive-rewarding mechanism to stimulate activities in social networking services. *International Journal of Network Management 22*, 1 (2012), 1–11.

Ognjen Scekic (oscekic@dsg.tuwien.ac.at) is a Ph.D. student in the Vienna PhD School of Informatics and the Distributed Systems Group at the Vienna University of Technology, Vienna, Austria.

Hong-Linh Truong (truong@dsg.tuwien.ac.at) is a senior researcher in the Distributed Systems Group at the Vienna University of Technology, Vienna, Austria.

Schahram Dustdar (dustdar@dsg.tuwien.ac.at) is a full professor and director of the Distributed Systems Group at the Vienna University of Technology, Vienna, Austria.

DOI:10.1145/2461256.2461276

How to mitigate a cyber-physical attack that disables the transportation network and releases a cloud of chlorine gas.

BY NABIL ADAM, RANDY STILES, ANDREW ZIMDARS, RYAN TIMMONS, JACKIE LEUNG, GREG STACHNICK, JEFF MERRICK, ROBERT COOP, VADIM SLAVIN, TANYA KRUGLIKOV, JOHN GALMICHE, AND SHARAD MEHROTRA

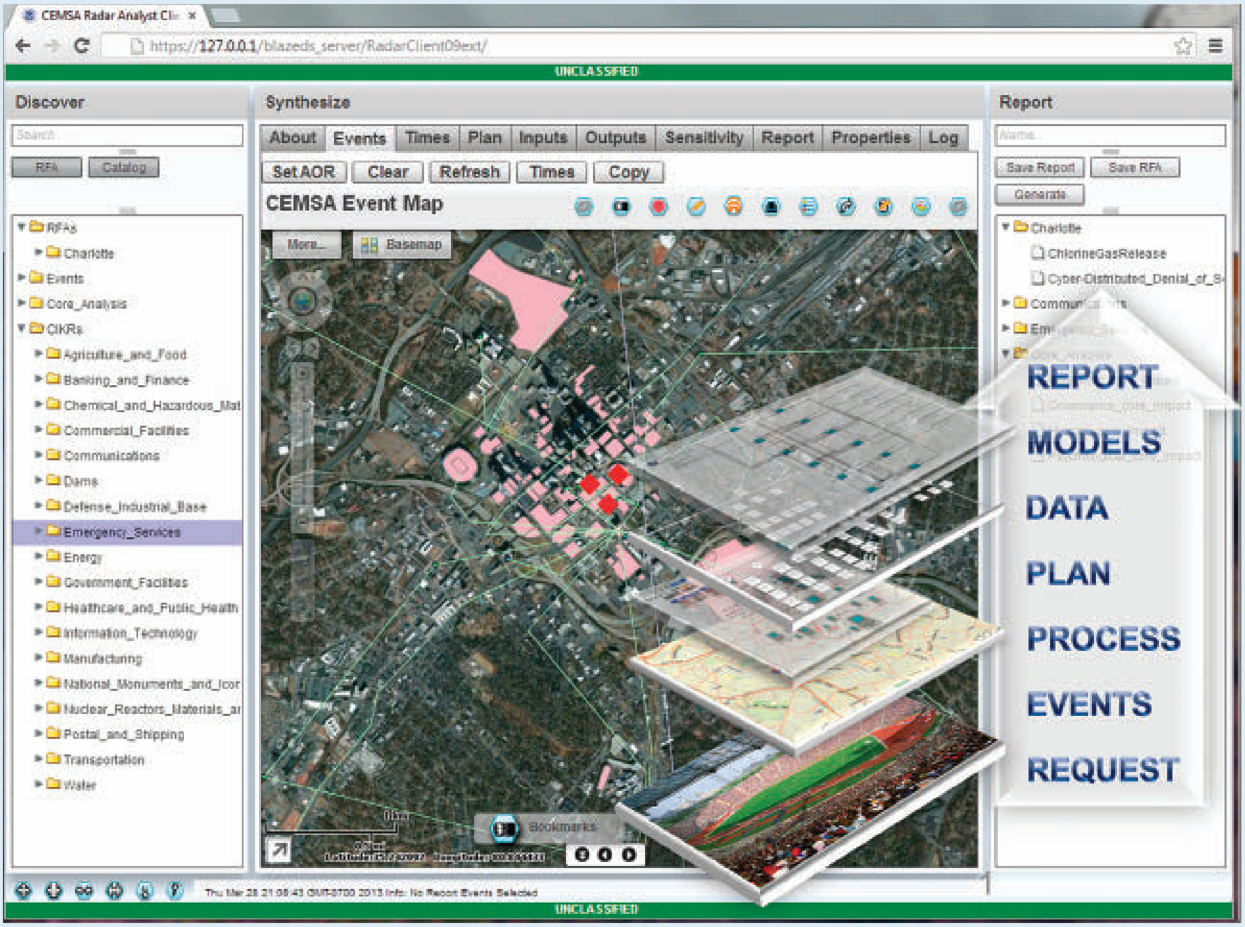
Consequence Analysis of Complex Events on Critical U.S. Infrastructure

U.S. DEPARTMENT OF Homeland Security analysts develop simulation models and tools to analyze the consequences of complex events on critical U.S. infrastructure and resources. An example of such an event is a coordinated cyber/physical attack that disables transportation and causes the release of a toxic chemical plume. The results can inform policymakers at the local, state, regional, and national levels. The Complex Event Modeling, Simulation, and Analysis, or CEMSA, program in the DHS Science and Technology Directorate is developing and deploying such a system to let analysts quickly integrate data, models, and expertise to arrive at credible consequence analysis of complex events. CEMSA aims to reduce turnaround time and costs, provide

» key insights

- **DHS analysts must perform their assessments quickly and therefore require tools that support quick response.**
- **CEMSA reduces the time analysts need for initial assessments while expanding the scope of simulations.**
- **As in any automated system, visibility is needed into where underlying data and models are available for inspection and modification.**

Figure 1. CEMSA enables rapid analysis of complex event consequences.



organic capabilities for risk analysis within DHS, enhance interoperability within DHS, and enable DHS to access and leverage the best available models within other government agencies, as well as within partner universities and industry.

Here, we start with the complex event analysis environment, followed by an approach to addressing them. We briefly present technical detail of some CEMSA components and discuss an example of its possible use in an interesting homeland security application—an evaluation of the consequences of cyber events on the physical infrastructure (see Figure 1).

Complex Event Analysis

CEMSA is being developed for an operational environment to support DHS strategic- and operational-level planners. Products and services in crisis response require analysis to be conducted within given time constraints to meet DHS leadership decision cycles. CEMSA addresses the following requirements:

Analytical.

- Enables estimation of disruption consequences, guides use and development of more detailed models, and integrates models;
- Allows composition of current and future models in an operational environment to determine direct and cascading effects resulting from multiple sources of infrastructure disruption;
- Enables analysis and simplification of systemwide models, allowing estimation of disruption consequences, guides use and development of more detailed models, and assists development of high levels of confidence in model results;
- Ensures transparency for clarity and ease of communication, enabling explanation of possible system behaviors;
- Confirms modeling and analysis sufficient to engender a high level of confidence in modeling and analysis results; and
- Quantifies errors and uncertainty that can arise when combining real-time data streams with models based on historical data or analytic abstractions.

Real time. CMESA enables real-time decision making through the collection and management of real-time field information, identify effects of new information on an analysis already under way, and project effects of new information, with updates from the field on a simulation's outcomes.

Architectural.

- Links models across and within infrastructures at various resolutions and spatial and timescales while accommodating various modeling languages and approaches;
- Enables on-the-fly integration of multiple, disparate models incorporating consequence and other analyses to address specific questions and provide

analytical ability scaled to available schedule and budget; and

- Applies well-defined "semantics" for describing models and simulations, leveraging and expanding the existing suite of analytical tools and capabilities and developing methods to incorporate existing and potential new tools.

Standards. CEMSA adheres to relevant industry standards, including those from the World Wide Web Consortium and from the Open GeoSpatial Consortium.

Approach

The nonproprietary, net-centric, enterprisewide CEMSA[a] delivers innovative capabilities for addressing these requirements: It is designed and implemented as a modular system based on open service-oriented architecture standards. A number of modular core CEMSA Spring Framework services are implemented on the back-end server to support consequence analysis (see Figure 2). The back-end server includes an execution-engine service based on Kepler/Ptolemy software[5] that supports combined execution approaches, a catalog that provides knowledge reasoning about models and simulation domains, a central database service based on PostgreSQL for persistence, an approximation engine for delivering initial time-critical analysis results, a planning service that generates candidate simulation plans for analysts, and an explanation engine supporting assessment of the results.

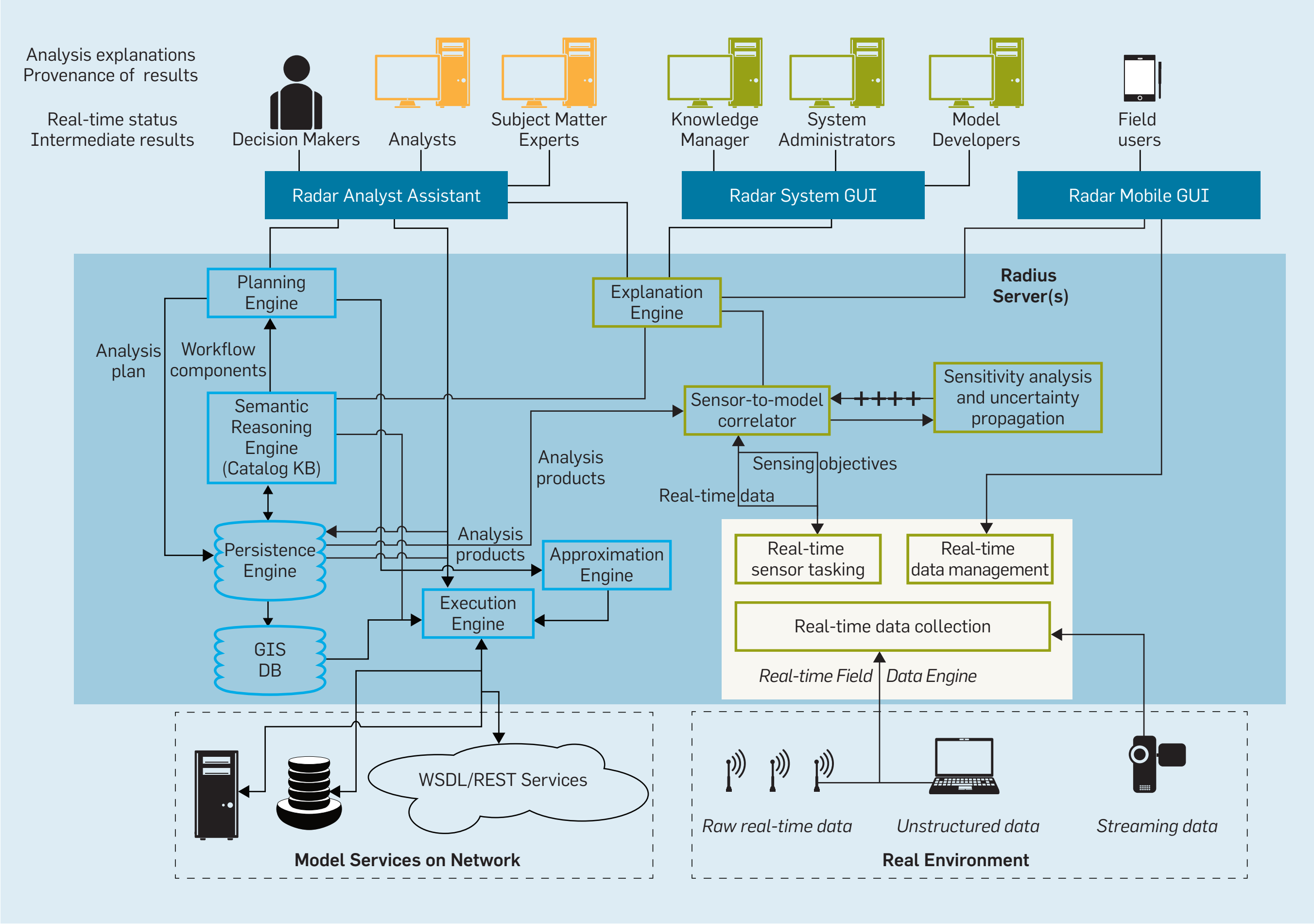
Net-centric analyst assistant. The Web-based CEMSA user interface helps analysts develop, manage, and assess the effects of multiple complex events, supporting their requests for analysis (RFA), delivering a consequence analysis report, and communicating with CEMSA back-end services for model composition.

The CEMSA front-end is based on a model-view-controller approach. Panels are laid out from left to right for overall analysis tasks of discovery, synthesis, and reporting. Discovery views include RFAs, data, models, experts, and ontology trees of Critical Infrastructure and Key Resources (CIKR), which are part of the United States National Response Framework. Synthesis views include events maps, inputs, outputs, sensitivity, plans, and system properties. Reporting views include report templates, generation options, and report editing.

When an analyst requests a plan, the constraints are passed to the planning engine, which returns a generated plan workflow for the analyst's review. After one or more iterations where the analyst identifies specific models, data, and experts, and receives refined generated plans, the analyst then executes the plan. Execution is coordinated by the execution engine, with calls to models on the network operating as services. The analyst can use the

[a] Developed by Lockheed Martin for the Department of Homeland Security, CEMSA is owned by DHS, with no Lockheed Martin copyright, and is free for DHS to install.

Figure 2. CEMSA architecture integrates models and data.



same view to monitor, pause, stop, or change plan execution. A report is generated when the consequence analysis simulation is completed.

Planning engine. The planning engine generates an analysis plan from a description of the business processes represented by the DHS functional area analysis, the DHS infrastructure data taxonomy, and catalog of models. This information is stored in the meta-plan, an XML representation of a directed acyclic graph of meta-models covering all required analysis activities.

The planning engine uses the meta-plan to generate a plan as a directed acyclic graph of meta-models that responds to the RFA. It generates a hierarchical task network, including a hierarchy of analysis activities, within which are active task networks that specify potential combinations of categories of models. The hierarchical task network approach is used in many industrial-strength planners to reduce the search space of potential plans to those corresponding to actual task activities in a given domain, in this case consequence analysis.[4]

The planning engine selects and composes resources that satisfy constraints (such as time, fidelity, and inclusion or exclusion of specific models, data, and experts used in generating a solution). Iteration through the CEMSA Radar user interface selects among alternative models to perform the actual model computations appropriate for that model category. The planning engine estimates the overall duration of the resulting plan, and if a time constraint cannot be met, alternate models are substituted into the plan to reduce the overall timeline. If the time constraint is still unmet, the planning engine uses model approximations. The user interface gives analysts the ability to modify the model composition and replace selected models with others as needed.

The planner translates the completed plan into an executor-appropriate format. For the Kepler executor, this is a workflow of "actors" representing Web Service Description Language (WSDL) calls and edges representing the data passing between models.

A disruption to supervisory control software might shut down an electrical generator, and the resulting loss of power might disable a water-pumping station.

Approximation engine. The approximation engine enables timely consequence analysis through estimates of analysis plan outputs and of end-to-end run times as a function of desired granularity. It characterizes the uncertainty associated with the models and approximates individual models with a simpler surrogate model. The approximation engine works with the planning engine by applying statistical analysis to similar previous analysis plans, captured in a knowledgebase, to provide an approximate answer. This approach is similar to the one used in engineering optimization, where model approximations (surrogates) are used for repeated runs of the composite simulation.[1]

CEMSA's simulation computes probability distributions for desired parameters describing complex models consisting of existing models. Parallel and distributed simulations, in which multiple simulations consisting of individual models or simulations execute simultaneously on different processors/platforms, provide the best scalability as requested analyses become more complex and more users interact with the system.

Semantic reasoning engine. A key challenge for model composition is understanding when models can be coupled together and what models can be coupled as part of a larger simulation; this requires capturing and reasoning over model type information, as well as model input and output types. CEMSA's semantic reasoning engine addresses this semantic challenge, with the catalog component of the engine serving as a knowledgebase and reasoner storing previous studies, interdependencies, models, simulations, and datasets.

Ontologies in the catalog support model composition in the planner, categorizing models and their inputs. CEMSA's base ontologies are Semantic Annotations for WSDL Working Group and OWL-S ontology built on top of the Web Ontology Language (OWL) for services, Kepler for model composition, and an adaptation of the DHS infrastructure data taxonomy; the table here summarizes the standards governing CEMSA's catalog representation.

When the planning engine populates an abstract task in the hierarchical task network, it identifies context, including event type, CIKR sectors, and activity the analyst requested. The catalog iterates through contexts to identify the models that are relevant to the event, sector, and activity, using an ontological reasoner to check for satisfaction over the class hierarchy in the ontologies.

System administrators integrating new models and simulations into the CEMSA framework modify the catalog through the knowledge-manager-perspective user interface that supports describing how a class of models fits into the analysis process; modelers provide details for model instances through a modeler-perspective user interface for describing the model itself. The planning engine in turn requests types of models matching constraints from the catalog, as do the approximator engine and the analyst assistant when presenting these constraints for selection by the analyst.

Model ingestion engine. The primary requirement for adding models is that they provide an external API from which developers can construct a Web service. CEMSA represents service interfaces through WSDL, invoking them through Simple Object Access Protocol (SOAP). Modelers construct their own Web service wrappers or use the facilities provided by the Kepler scientific computing framework, which also serves as the basis for CEMSA's execution engine.

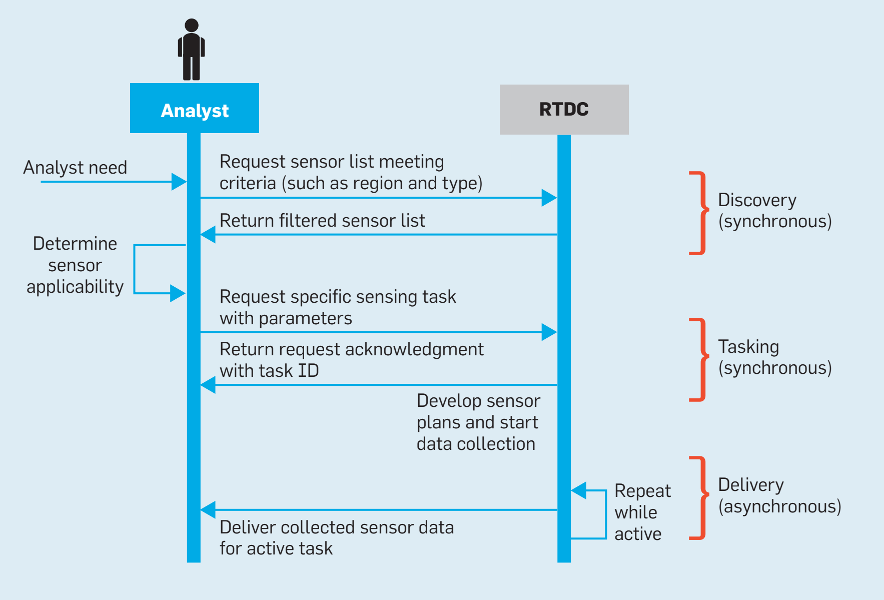
Once models are wrapped as a WSDL, the modeler can ingest them into the CEMSA catalog through the Radar user interface. Using the catalog, the model-ingest client prompts the modeler to provide details (such as execution time, units, and types used on inputs and outputs and valid domains for the model), as well as information on standards in the table. When the modeler supplies the metadata, the ingest client stores the Kepler wrapper and metadata in the catalog.

Knowledge manager assistant. A knowledge manager (KM) examines the CEMSA catalog of models, data services, and experts, as well as the end-user organization's analysis process, and provides the planning engine task hierarchies that fit the organization's business processes. The meta-plan states the sequencing of overall activities and potential couplings between categories of models. CEMSA provides a perspective for the KM to achieve these goals.

Modeling standards used by CEMSA.

Category	Standard	Description
Measures	Metric	Metric measures (meters, liters, grams) for model inputs and outputs and seconds for temporal measures
Network Protocol	SOAP 1.2 over HTTPS	Network transport uses SOAP xml over https for secure data transmission
Services	WSDL 1.1	Model provides callable input, output, and control interface through the WSDL 1.1 standard
Semantic Models	OWL-DL	Describes inputs and outputs for models that provide semantic results or use semantic inputs
Queries	SQL, SPARQL	SQL database queries from model to CEMSA database service; SPARQL queries from model to CEMSA ontology knowledgebase service
Information Exchange	NIEM	CEMSA emphasizes core, geospatial, CBRN, cybersecurity (emerging), emergency management, and infrastructure protection sectors of the National Information Exchange Model
Geospatial Data	OGC standards: KML, WFS 2.0, WMS 1.3, WCS 2.0, WPS 1.0	Support standards include KML for 3D overlays, annotations and interaction; Web Feature Service for map features; Web Mapping Service for images, Web Coverage Service for grid data of geospatial regions; and Web Processing Service for algorithms or processes operating on geospatial data, including process status
Geographical Coordinate System	WGS 84	Uniform coordinate system reduces processing time and potential translation errors
Control	CEMSA	Service interfaces to models include initialization, time advance, play, pause, resume, and stop
Test Cases	CEMSA	Provide example inputs, outputs, ranges, and datasets for testing and characterizing model operation

Figure 3. Real-time data can be combined with simulations.

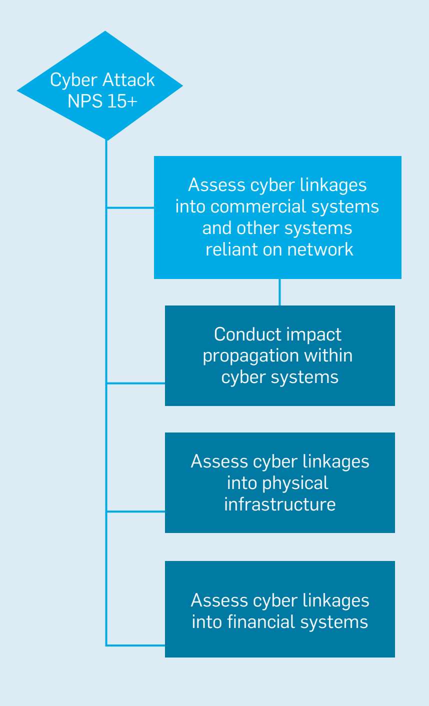

Analyst
RTDC
Analyst need
Request sensor list meeting criteria (such as region and type)
Return filtered sensor list
Discovery (synchronous)
Determine sensor applicability
Request specific sensing task with parameters
Return request acknowledgment with task ID
Tasking (synchronous)
Develop sensor plans and start data collection
Repeat while active
Deliver collected sensor data for active task
Delivery (asynchronous)

This perspective can display all model information in the catalog. The KM examines the model metadata, aligns the model with the appropriate analysis phase or activity, and edits the meta-plan to assign possible model input/output connections. The KM can commit the meta-plan for future use from this perspective. The perspective also provides access to previous model results and performance data as another resource for updating the meta-plan.

Real-time field data engine. The real-time field data engine enables analysts to use near-real-time information from third-party sensors that feed information into the CEMSA system; the real-time sensor information is a way to inform model compositions in CEMSA.

The real-time data collection (RTDC) module extends the CEMSA architecture with an integrated data-collection mechanism for real-time access to heterogeneous sensing and live data input mechanisms. The functionality of the RTDC is divided into three distinct steps (see Figure 3): The first is "sensor discovery," or the process of identifying relevant sensors to probe for the event at hand. The second is "sensor tasking," where sensors are activated with data-collection plans to match resource needs. Sensor-tasking requests come in a variety of types: periodic, where sensor data is scheduled to arrive periodically; event-based, where sensor data arrives upon occurrence of an event (as determined by an event condition); instantaneous one-off, where the `model_to_sensor_correlator` service generates a one-time sensor data demand that could arrive at any moment. Any of these requests may be deadline-based where the result of the sensing is required within a deadline based on when the request is made. The final step is sensor data delivery.

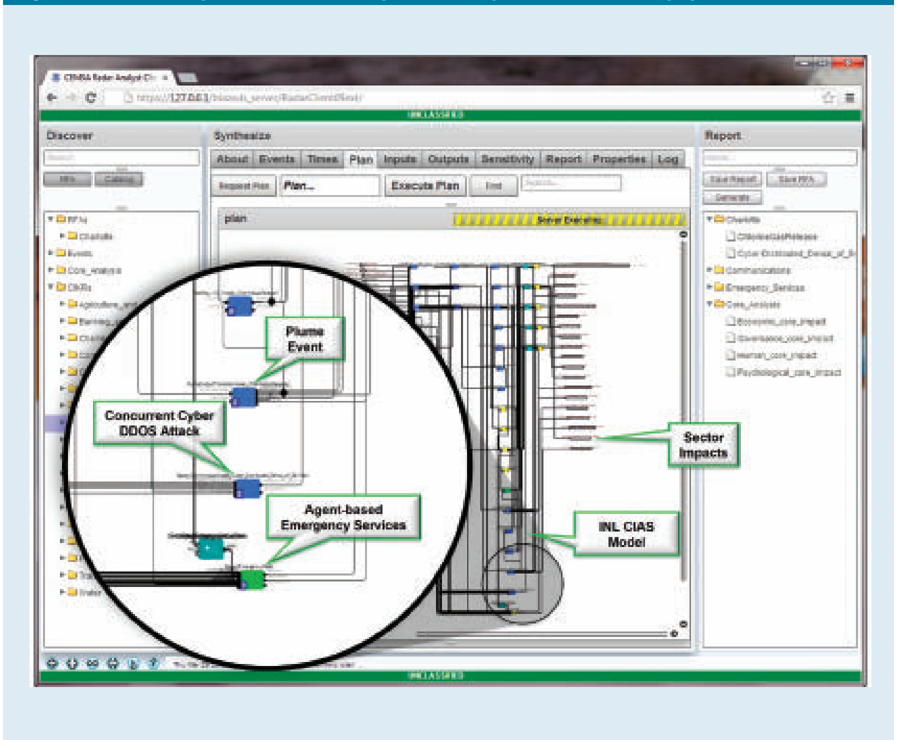
Figure 4. Functional area analysis defines plans, including cyber effects.


Cyber Attack
NPS 15+
Assess cyber linkages into commercial systems and other systems reliant on network
Conduct impact propagation within cyber systems
Assess cyber linkages into physical infrastructure
Assess cyber linkages into financial systems

The RTDC design is based on the SATware middleware[3] for sensor-based systems that support high-level abstraction of sensors and sensor resources to abstract the heterogeneity and diversity of sensors and sensor platforms. Such heterogeneities make programming pervasive applications highly complex, especially when applications must explicitly deal with failures, disruptions, timeliness properties under diverse networking and system conditions, and missing or partial information. In RTDC, applications deal with higher-level semantic concepts (such as temperature, traffic level, and presence of entities/activities) at a given location and time. Such a semantic abstraction is provided through "virtual sensors" bridging application-level concepts and raw sensor data using "operators" that transform input sensor data streams (such as weather sensors) to higher-level semantic streams that capture application-level concepts and entities (such as region affected by hurricane). A phenomenon may be observed through multiple sensors by defining appropriate virtual sensors and/or combining inputs from multiple virtual sensors through the sensor composition language of SATware. Virtual sensors, when defined and registered with RTDC, hide the complexity of sensor programming from applications.

To address the challenge of real-time acquisition and processing of sensor data, RTDC models the sensor-data-processing problem as a constrained optimization problem.[7] Applications have associated with them a benefit function that models the utility of observing a phenomenon to the application. With CEMSA, that benefit may correspond to reduced uncertainty in the phenomenon of interest. The sensor-acquisition task is subject to constraints (such as artifacts of the sensor properties, as when a given camera can be focused on only one region at a given time), the acquisition process (such as a website where data is acquired and might impose restrictions on the number of queries an application can pose per unit time), and resource constraints (such as network bandwidth or limitations of available computational resources). Moreover, sensors involve associated actuation parameters (such as spatial/temporal resolution of data capture and other quality measures like errors/deviation from the actual value). RTDC chooses actuation parameters that maximize the expected benefits while ensuring the constraints imposed by the sensors, resources, and collection process are satisfied.[7]

Figure 5. Generated plans model consequences of cyber events on the physical infrastructure.



Explanation engine. The explanation engine helps analysts understand the provenance and applicability of results by associating data and model results, collecting simulation data at the time of analysis, and presenting results. It supports real-time decision making by tracing where new data caused an analysis workflow to diverge from expected or previous results. The explanation engine stores data from throughout the composition and execution life cycle, including starting data, intermediate values, and final values. Analysts search across intermediate and final results for each run and use persistent data from prior runs to compare results.

The explanation engine lets analysts visually trace the running execution and the simulation results to an individual model and its associated inputs and outputs through an editable data-flow diagram in the plan view, providing error bounds for final results as a function of error bounds of each individual model. Sensitivity analysis is a key method for explaining data and model results. Starting with a plan from the planner, the explanation engine uses Monte Carlo methods and model approximations to provide input to a clone of a given plan, simulate its run, and calculate the distribution across each numeric output value.

Assessing Consequences of Cyber on the Physical

Now consider the following scenario. Preparations for a major public event held in a major metropolitan area (such as a national political party's convention) include analysis of the consequences of an attack on infrastructure. Analysts would use CEMSA to generate an analysis plan in response to an RFA. CEMSA generates the plan from a description of the underlying process represented in the DHS functional area analysis (see Figure 4).

One scenario analysts plan for is a distributed denial of service (DDoS) attack timed with highway explosions releasing toxic chlorine gas upwind of the political event, a scenario that represents a "complex event" with concurrent and cascading effects. The DDoS attack is under way while the explosion and release of the gas occurs (concurrent disruptions), with each of these initial events producing cascading effects on multiple infrastructure sectors.

Effects on communications network QoS degrade the capacity and reliability of communications between citizens and responders, causing dispatch delays and errors that lengthen response times and increase casualty rates.

The DDoS attack impairs communications by citizens, emergency responders, and industrial-control systems required for the safe operation of electrical-, water-, petroleum-distribution, and other networks. The explosion and gas release disrupt the regional transportation grid, and the ensuing plume requires citizens to shelter in place while causing illnesses that require health-care and emergency medical services. Both events interact with the load already imposed on local and regional assets by the political event in the form of concentrated demand for communications, information technology, and transportation resources, as well as additional responsibilities for police and emergency managers. The consequences of these interacting events ripple across the region.

An analyst selects a subset of interesting features from these possible interactions and affected CIKR sectors; for example, the CEMSA team has evaluated the performance of emergency responders—their ability to promptly answer 911 calls triggered by the interacting events, in addition to the base load of police, fire, and medical calls—as affected by changes in multiple network sectors (such as communication, water, and electrical power). In this case, the planning engine generates an analysis plan that, in addition to physical models, includes multiple cyber impact models spanning multiple infrastructure sectors and using different simulation techniques and models of computation (see Figure 5).

Infrastructure networks model. The Critical Infrastructure Analysis and Simulation (CIAS)[b] model performs flow-graph analysis and simulation of networked infrastructure, including water (such as pumping stations, pipelines, and consumers), power (such as generators, transmission, and distribution lines and loads), and communications (such as network exchanges, backhaul, and access points). Users specify interdependencies among infrastructure assets by setting attributes for individual as-

[b] Developed by the Idaho National Laboratory, Idaho Falls, ID.

sets (such as minimum demand and maximum capacity for commodities like water and electricity).

CIAS performs flow and reachability analysis to propagate effects among infrastructure networks stored in a geospatial information system's database. Reachability analysis determines the existence of source-to-sink paths and essential links for each infrastructure sector. Flow analysis computes the flow of resources (such as water volume and data throughput) available at each point, recording shutdowns and impairments when flow levels are below minimum requirements for assets.

Interdependencies among the supported sectors propagate failure modes across networks; for example, a disruption to supervisory control software might shut down an electrical generator, and the resulting loss of power might disable a water-pumping station. A nearby communications exchange might have backup power, but loss of cooling water will cause it to shut down. The resulting loss of network connectivity might require a nearby water-treatment plant to shut down. Effects propagate to other sectors that depend on networked commodities (such as emergency managers who require network connectivity to dispatch 911 calls and health-care providers who require water and electricity to run an emergency room).

Emergency services model (ESM). Inspired by Mysore et al.,[6] the ESM[c] is an "agent-based" simulation that computes "network infrastructure" effects (particularly impairment to transportation and communications networks) on the capacity and efficacy of police, fire, and emergency medical services. It represents individual health-care and emergency-services assets and citizens as agents, or computational entities responding to the simulated environment and other agents by following simplified rules that reflect essential behaviors.

The ESM is implemented in the Repast Simphony suite[d] and loads transportation network information from the OpenStreetMaps dataset, locations of emergency services, and health-care and communications assets from the Homeland Security Infrastructure Program (HSIP) Gold[e] 2012 dataset. Like CIAS, the ESM extracts "network" relationships from geospatial data. The datasets organize data into "layers" according to asset type (such as roads, law-enforcement facilities, and cell towers), describing each instance of an asset by a "shape"; for example, the road network consists of a set of road segments (represented in the dataset by polylines) that meet at intersections specified by (latitude, longitude) points. The ESM converts this spatial representation into an undirected graph with vertices corresponding to intersections and edges corresponding to road segments, using the graph representation to dispatch emergency responders along the shortest available path to the location of a casualty. The ESM uses heuristics to reconstruct connected networks from the sometimes-disconnected segments in geospatial datasets; for example, it "enlarges" the road segments' endpoints to connect them to adjacent segments, as when two or more multi-lane roads meet at an intersection that could be tens of meters wide.

The HSIP Gold dataset provides limited information about core network assets and backhaul links maintained by commercial voice and data carriers, so the ESM uses a spanning-tree approximation to connect "edge" assets like cell towers to core assets like switching centers. Each mobile responder is affiliated with a fixed site—police cars with police stations, fire engines with fire stations, and ambulances with hospitals.

Each asset type follows a simple state machine: citizens can be `healthy`, `ill`, `injured`, or `dead`; fixed sites can be `available`, `limited`, or `unavailable`; and mobile responders can be `available`, `dispatched`, `on_scene`, `limited`, or `unavailable`. Each asset type can

The scenario envisions a DDoS attack on the carrier's 4G network, in which the communications model used by DEMSA simulates the cell towers closest to the political event and deploys multiple users in each cell to measure network impairment caused by the attack.

[c] Developed by the CEMSA team.

[d] Developed by the University of Chicago and Argonne National Laboratory.

[e] DHS HSIP Gold is a unified homeland infrastructure foundational geospatial data inventory assembled by the National Geospatial Intelligence Agency in partnership with the Department of Defense, DHS, and the U.S. Geological Survey for use by the homeland security/homeland defense community; it includes a compilation of best available federal government and commercial proprietary datasets.

enter the `limited` or `unavailable` state to model the effects of resource rationing and starvation as computed by CIAS, with communications between assets suffering quality-of-service (QoS) impairment computed by CIAS and the OpNet communications sector model (http://www.opnet.com/).

The ESM provides concrete examples of the consequences of impairment to networked infrastructure caused by cyber events. Effects on communications network QoS degrade the capacity and reliability of communications between citizens and responders, causing dispatch delays and errors that lengthen response times and increase casualty rates.

OpNet long-term evolution model. The OpNet modeler is a commercial modeling and simulation tool for voice and data networks. It is a discrete-event simulation in which "models" corresponding to software applications, networking protocols, and hardware devices interact by exchanging messages corresponding to networked data. OpNet and its partners have implemented several model libraries, including a wireless library with models of generic wireless devices (such as 802.11 Wi-Fi routers and terminals) and a long-term evolution (LTE) library that models the Third Generation Partnership Project (3GPP, http://www.3gpp.org/) LTE air interface. Subject-matter experts can use OpNet to simulate a range of use cases, including civilian, public-service, and military wireline and wireless networks carrying voice and data traffic.

Our scenario involving release of chlorine gas on a road, focuses on a region with a dominant commercial carrier that provides 3G (using code division multiple access, or CDMA, waveforms) and 4G (using the 3GPP LTE waveform) voice and data services to civilian and public-service users. It focuses on the 4G LTE network, which offers the highest end-user data rates and therefore drives the architecture of the carrier core network. LTE is an all-packet-switched, all-IP network that treats voice messages as a particular class of packet data. This converged approach, along with improvements in modulation and multiple access protocols, significantly increases network capacity. It also involves the possibility that abuse of the network by one class of application could impair the performance of others by exhausting shared resources. The scenario envisions such an event, in the form of a DDoS attack on the carrier's 4G network, in which the communications model used by CEMSA simulates the cell towers closest to the political event and deploys multiple users in each cell to measure network impairment caused by the attack. Each simulated user runs three simulated applications: voice (periodically initiating calls of varying duration), text messaging, and background IP data.

Related Work

The High Level Architecture (HLA) is a specification developed by the U.S. Department of Defense Modeling and Simulation Office with several implementations, including the MaK Run-Time Infrastructure. It emphasizes information exchange between simulations over an information bus at the syntactic level using the Federation Object Model. Model composition in HLA is highly dependent on the selected run-time infrastructure (RTI). Composing models based on RTIs usually means implementing a custom gateway between RTIs. HLA results in a static architecture for model federation that is difficult to change dynamically, as in the CEMSA planner.

Conclusion

We have described the CEMSA system and the algorithms being developed to initial operating capability. The CEMSA approach is based on semantic model composition via hierarchical task network planning. Applying constructs from the planning community and the engineering-optimization community to infrastructure impact analysis, CEMSA gives analysts the means to assemble, coordinate, and evaluate collections of models for complex events and quickly arrive at effective decisions.

Acknowledgment

This work was funded by the Infrastructure Protection and Disaster Management Division, Science and Technology Directorate, Department of Homeland Security, Washington, D.C.

References

1. Eldred, M.S., Adams, B.M. et al. *DAKOTA: A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 5 Reference Manual.* Sandia Technical Report SAND2010-202184 (Version 5.2), Nov. 2011; http://dakota.sandia.gov/docs/dakota/5.0/Users-5.0.pdf
2. Homeland Security Infrastructure Program Gold Dataset; http://www.dhs.gov/infrastructure-information-partnerships
3. Hore, B., Jafarpour, H., Jain, R., Ji, S., Massaguer, D., Mehrotra, S., Venkatasubramanian, N., and Westerman, U. Design and implementation of a middleware for sentient spaces. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics* (New Brunswick, NJ, May 23–24). IEEE, 2007, 137–144.
4. Kambhampati, S. A comparative analysis of partial order planning and task reduction planning. *SIGART Bulletin 6*, 1 (Jan. 1995), 16–25.
5. Ludascher, B. et al. Scientific workflow management and the Kepler system: Research articles. *Concurrent Computing: Practice and Experience 18*, 10 (Aug. 25, 2006), 1039–1065.
6. Mysore, V., Narzisi, G., Nelson L., Rekow, D., Triola, M., Shapiro, A., Coleman, C., Gill, O., Daruwala, R.-S., and Mishra, B. Agent modeling of a sarin attack in Manhattan. In *Proceedings of the First ACM International Workshop on Agent Technology for Disaster Management* (Hokkaido, Japan, May 8–12). ACM Press, New York, 2006, 108–115.
7. Vaisenberg, R. *Towards Adaptation in Sentient Spaces.* Ph.D. Dissertation, University of California, Irvine, 2012; http://www.ics.uci.edu/~ronen/resources/thesis.pdf

Nabil Adam (adam@adam.rutgers.edu) is a Distinguished Professor of computer and information systems and founding director of the Center for Information Management, Integration and Connectivity at Rutgers University; he was the CEMSA program manager while serving as a fellow and senior program manager in the Science and Technology Directorate of the U.S. Department of Homeland Security, Washington, D.C.

Randy Stiles (randy.stiles@lmco.com) is a senior staff research scientist at the LM Advanced Technology Center in Palo Alto, CA, and LM program manager for CEMSA.

Andrew Zimdars (andrew.zimdars@lmco.com) is a staff research scientist at the LM Advanced Technology Center, Palo Alto, CA, and led the CEMSA Real-time Analysis Communication Environment project.

Ryan Timmons (ryan.p.timmons@lmco.com) is a senior research scientist at the Lockheed Martin Advanced Technology Center, Palo Alto, CA.

Jackie (Man-Kit) Leung (jackie.leung@lmco.com) is a research scientist at the Lockheed Martin Space System Company, Palo Alto, CA.

Greg Stachnick (gstachni@comcast.net) is a principal software engineer at LM Integrated Systems & Global Solutions Advanced Technology Office, San Jose, CA, where he led development of the CEMSA planning service.

Jeff Merrick (jeff.r.merrick@gmail.com) was a firmware and software engineer for the CEMSA planner at Lockheed Martin Integrated Systems & Global Solutions, San Jose, CA.

Robert Coop (Robert.Coop@rtsync.com) is a software developer at the RTSync Corporation, Knoxville, TN.

Vadim Slavin (vadim.a.slavin@lmco.com) is a software engineer, research scientist, and entrepreneur in Palo Alto, CA.

Tanya Kruglikov (tanya.s.kruglikov@lmco.com) is a senior software engineer at IBM Research - Almaden, San Jose, CA.

John Galmiche (john.galmiche@gmail.com) is an associate at Booz Allen Hamilton, Arlington, VA.

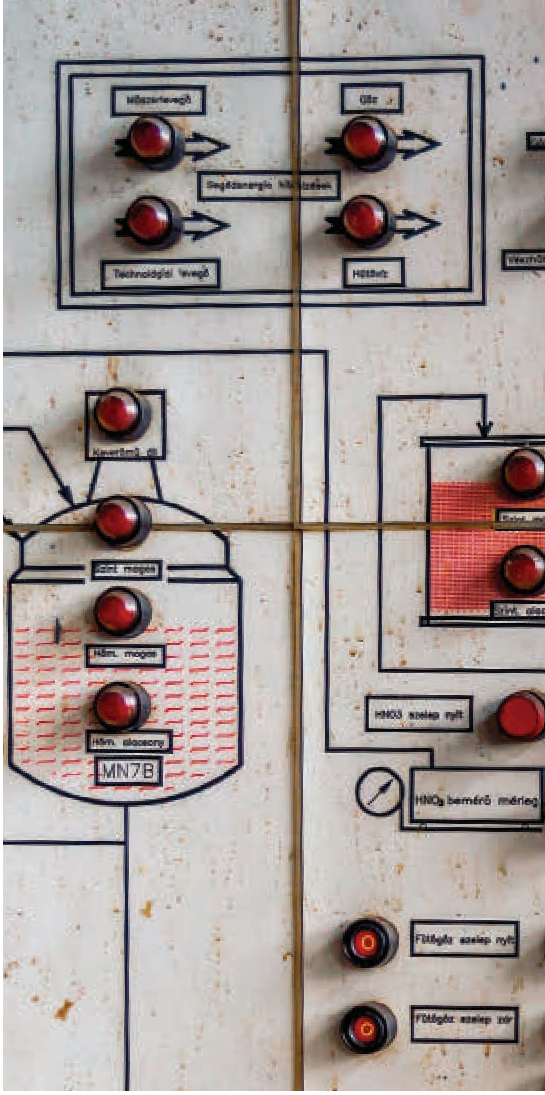
Sharad Mehrotra (sharad@ics.uci.edu) is a professor in the Department of Computer Science and founding director of the Center for Emergency Response Technologies at the University of California, Irvine.

DOI:10.1145/2461256.2461277

How to offer recommendations to users when they have not specified what they want.

BY DEEPAK AGARWAL, BEE-CHUNG CHEN, PRADHEEP ELANGO, AND RAGHU RAMAKRISHNAN

Content Recommendation on Web Portals

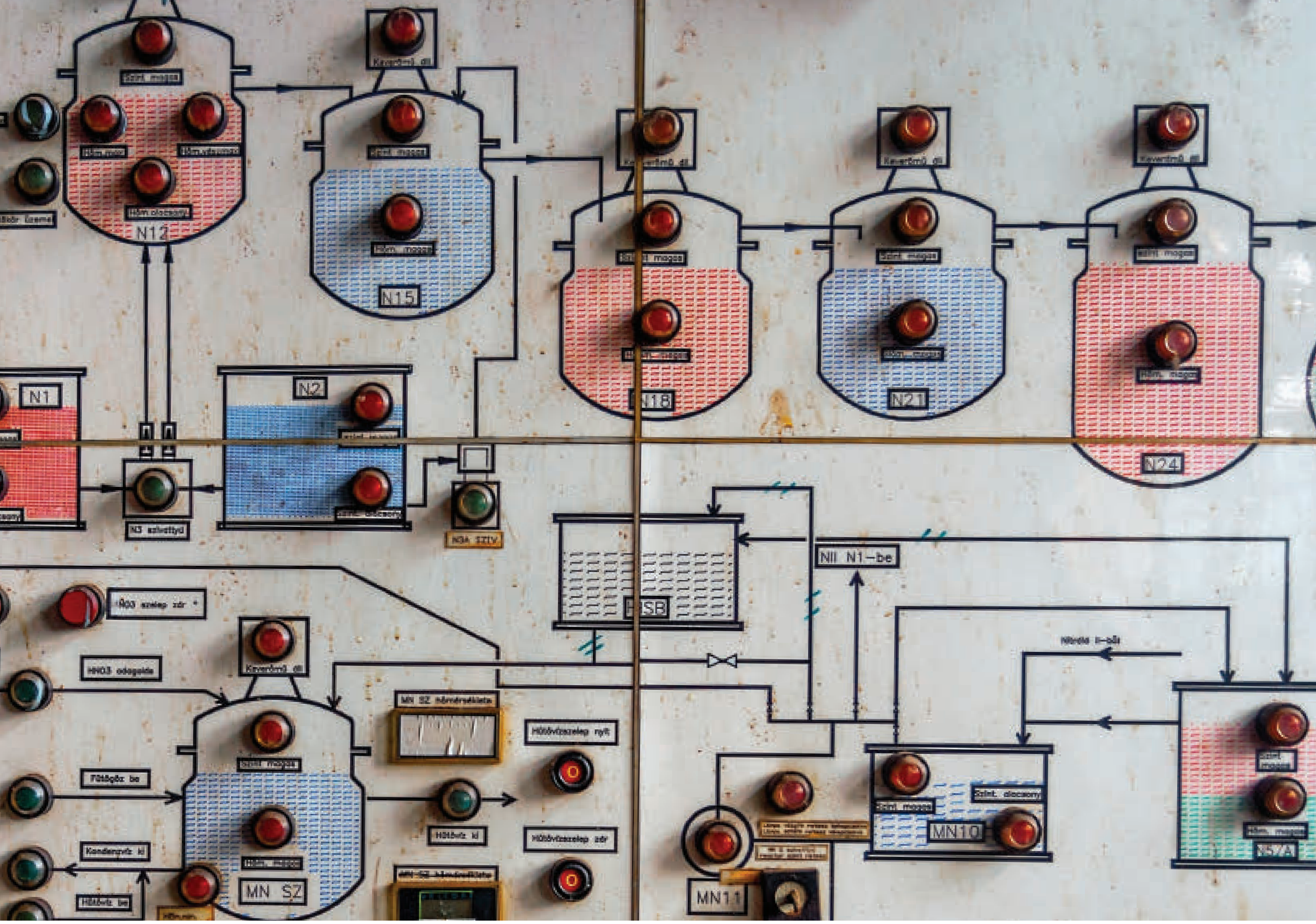


INFORMATION DISCOVERY HAS been transformed by the Web, and the impact on how information is gathered, published, and delivered has been profound. Web search is one form of discovery, and is preferred when a user has a specific objective, for example, wants directions to an address. It is less effective when users simply want to be informed about news that may be relevant to them, or to learn more about topics of interest. In these latter scenarios, the user experience depends crucially upon the quality of content recommendations. In contrast to search, the explicit signal about what the user wishes to see is much weaker, and the importance of a broad range of complementary indicators increases greatly. Novel techniques are required to best leverage a broad array of weak signals.

In this article, we present an overview of the content recommendation problem, namely how to recommend a small set of items to a user from an underlying pool of content items, in settings where the user does not explicitly express what is desired. In contrast to traditional media such as newspapers, every item shown on a page can be dynamically selected to reflect which items are currently the most engaging. In fact, the choice can be made taking into account a given user's interests, and even what they are looking at in the current session. Also,

» key insights

- Users search when they are looking for something specific; content recommendation techniques come into play when users are looking to be informed of news relevant to them, or browsing topics they are interested in.
- Unlike search, where the user's query is a strong indication of what they want, we must rely on a number of complementary indicators such as a user's long-term history, recent browsing behavior, and trends (popular topics) to identify the best content to recommend.
- There are a number of objectives that sites seek to optimize, including higher click-through rates, greater social interactions such as sharing of articles, and higher revenue or time spent on the site.



data-driven dashboards allow editors to program new stories to reflect trending topics and changes in the demographics of users visiting the site, and to monitor performance metrics in near real time. These changes are fundamentally altering how websites are designed, and indeed, how journalists and editors work.

The following issues must be considered in addressing the content recommendation problem:

Input signals. In building machine-learned models of what items a user is likely to engage with in a given context, we can draw upon many signals, including the content and source of each article, a user's interest profile (reflecting both long-term interests based on prior visits and short-term interests as reflected in the current session), and "popularity" indicators such as observed click-through rates or CTRs (the fraction of time the item is clicked on when a link to it is presented to users) and extent of social-sharing (for example, the number of times the item is tweeted or shared or "liked").

Objective to optimize. There are many objectives a website could choose to optimize for, including near-term objectives such as CTR and revenue per click, as well as long-term metrics such as increased time spent on the site, higher return and user retention rates, increase in social actions, and many others.

Algorithmic techniques. Algorithms must be developed to address a number of tasks.

- *Content analysis:* Create item profiles (for example, feature vectors) that capture the content with high fidelity.
- *User profile modeling:* Create user profiles that reflect the items they are likely to consume.
- *Scoring:* Estimate the likely future "value," for different types of values (for example, CTRs, semantic relevance to the user's current goal, or expected revenue) of showing an item to a user in a given context (for example, the page the user is viewing, the device being used, the current location).
- *Ranking:* Select a ranked list of items to recommend so as to maximize the expected value of the chosen objective function.

Editorial tools: A complete recommendation framework must additionally provide tools for editors and journalists to: (1) observe in real time what items are most interesting (to which user segments on which parts of the website, at what times, from what locations, and on what devices), (2) quickly identify emerging trends in order to create additional content to meet these information needs, (3) constrain the algorithmic recommendations to follow guidelines associated with the site, and (4) tune the objective function and balance trade-offs (for example, maximize revenues while still keeping CTRs high and delivering personalized user experiences).

While creating user and item profiles and providing editorial tools are important aspects of content recommendation, we focus on scoring and ranking in this article. We argue that since user intent in content recommendation is significantly weaker compared to applications like Web search (where a user specified query serves as a strong signal

PHOTOGRAPH BY OLIVER SVED

of intent), expected click-through rate or CTR (appropriately weighted) is a more fundamental measure in scoring items for a user in a given context than semantic relevance. Also, ranking is not merely sorting items by scores—we must balance considerations like diversity (ensuring that users see a range of topics over time), serendipity (ensuring that we do not overfit our recommendations to the user, thereby limiting the discovery of new interests) and *editorial voice* (the general tone of the content associated with a given portal). Further, the objective function for ranking might be based on several criteria, for example, we may want to maximize the number of article shares while not sacrificing more than 10% of the achievable CTR.

Application Settings

In Table 1, we identify four types of portals: general, personal, domain-specific, and social network. General portals publish a broad range of content; the home page of a content network typically falls in this category. A personal portal allows a user to customize the page with desired content; a domain-specific portal publishes content related to a specific topic or domain, for example, sports; and a social network portal allows users to connect to other users and disseminate information through their network. Content modules published on these portals broadly fall into one of the following three categories:

- *Featured Modules (FMs):* These recommend "interesting" and recent content to users. Figure 1 shows examples of FMs on three different general portals. Such FMs show heterogeneous content with links to items hosted across the content network (for example, sports, finance, and so on), and serve as a distribution channel sending users to different properties (domain-specific sites in the portal). General portals also have a set of domain-specific FMs that only recommend items from specific domains. For example, Figure 2 shows three domain-specific FMs for news, sports and entertainment on www.msn.com. On personal portals, personalized FMs provide recommendations that match each user's interests. For example, Figure 3 shows a personalized FM called "News For You" on my.yahoo.com, which recommends content items from the RSS feeds a user subscribes to. General and domain-specific FMs can also be lightly personalized. The content recommendation algorithms

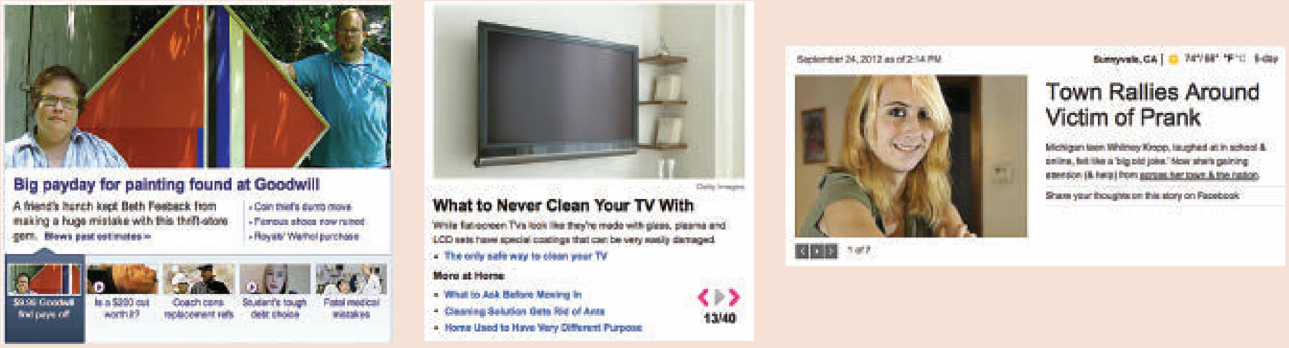
Table 1. Web portals and recommendation modules.

Portal Category	Examples	Typical Recommendation Modules	
General portal	www.yahoo.com www.msn.com www.aol.com	Home page	Featured module (FM: general) Featured module (FM: domain-specific)
Personal portal	my.yahoo.com igoogle.com my.msn.com	Home page	Featured module (FM: personalized)
Domain-specific portal	sport.yahoo.com	Home page	Featured module (FM: domain-specific)
	money.msn.com music.aol.com	Detail page	Related content module (RM)
Social network portal	facebook.com	Home page	Network update module (NM)
	linkedin.com twitter.com	Detail page	Related content module (RM)

Table 2. Available signals.

User	Are reliable user identifiers available? What user features (for example, demographics, location) are available?
Item	What is the size and quality of the pool of candidate items? What item features (for example, category, entities, keyword) are available?
Context	Is contextual information available? If so, what features of a context are available?
Feedback	Is user feedback (for example, clicks, ratings) available? How quickly can we use current feedback to update models?

Figure 1. General Featured Modules (FMs) on three different Web portals.



Big payday for painting found at Goodwill

A friend's hunch kept Beth Feeback from making a huge mistake with this thrift-store gem. Blows past estimates »

- Can thief's dumb move
- Famous shoes now ruined
- Royals' Warhol purchase

$9.99 Goodwill find pays off | Is a $200 cut worth it? | Coach cons replacement refs | Student's tough debt choice | Fatal medical mistakes

(a) www.yahoo.com

Getty Images

What to Never Clean Your TV With

While flat-screen TVs look like they're made with glass, plasma and LCD sets have special coatings that can be very easily damaged.

- The only safe way to clean your TV

More at Home:

- What to Ask Before Moving In
- Cleaning Solution Gets Rid of Ants
- Home Used to Have Very Different Purpose

13/40

(b) www.aol.com

September 24, 2012 as of 2:14 PM

Sunnyvale, CA | 74°/58° °F °C 5-day

Town Rallies Around Victim of Prank

Michigan teen Whitney Kropp, laughed at in school & online, felt like a 'big old joke.' Now she's gaining attention (& help) from across her town & the nation.

Share your thoughts on this story on Facebook

1 of 7

(c) www.msn.com

required for these three types of FMs have strong similarities.

▸ *Network-update Modules (NMs):* These recommend updates (that is, any information generated) from a user's neighbors in a social network. In contrast to FMs, items shown in NMs often reflect updates that are restricted to be seen only by connections of the user.[a] They include social actions like sharing, liking, and commenting. To make good recommendations, it is important to consider the reputation of the producer of an item, the strength of the connection between the producer and the recipient, and the nature of the social actions involved.

▸ *Related-content Modules (RMs):* These usually appear on a page whose primary content (for example, a news article) is the context, and recommend items "related" to that context. Figure 4 shows an example RM on huffingtonpost.com on the article "Mitt Romney health care plan wouldn't slow rising costs." In contrast to FMs and NMs, an RM has an additional piece of information: the context. Blending semantic relevance, item popularity, and how closely an item matches the user's interests usually makes good recommendations.

Content recommendation techniques are important and in use by almost all major Web portals. Instead of providing a review of specific techniques used by various portals, we review the broad framework that consists of two main technical components—scoring and ranking in various application settings. We believe this provides a crisp mathematical formulation of various use cases faced by Web portals in practice. The actual methods used can be adequately described by a combination of techniques within this framework.

Technical solutions to scoring and ranking in content recommendation modules depend on the goals of the application and the nature of available signals; Table 2 lists a number of typical scenarios. We start our review with a simple application setting, where the goal is to maximize clicks in a recommendation module that has a relatively small content pool, and makes

[a] Because of the private nature of NMs, we do not show screenshots.

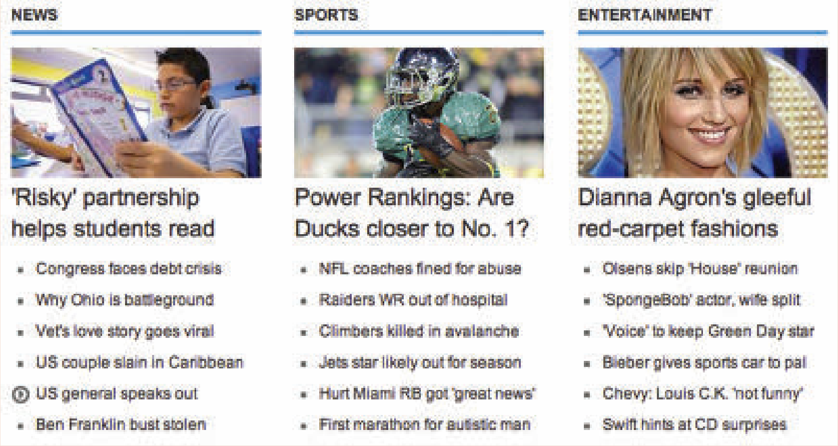


Figure 2. Domain-specific FMs on www.msn.com.

NEWS

'Risky' partnership helps students read

- Congress faces debt crisis
- Why Ohio is battleground
- Vet's love story goes viral
- US couple slain in Caribbean
- US general speaks out
- Ben Franklin bust stolen

SPORTS

Power Rankings: Are Ducks closer to No. 1?

- NFL coaches fined for abuse
- Raiders WR out of hospital
- Climbers killed in avalanche
- Jets star likely out for season
- Hurt Miami RB got 'great news'
- First marathon for autistic man

ENTERTAINMENT

Dianna Agron's gleeful red-carpet fashions

- Olsens skip 'House' reunion
- 'SpongeBob' actor, wife split
- 'Voice' to keep Green Day star
- Bieber gives sports car to pal
- Chevy: Louis C.K. 'not funny'
- Swift hints at CD surprises

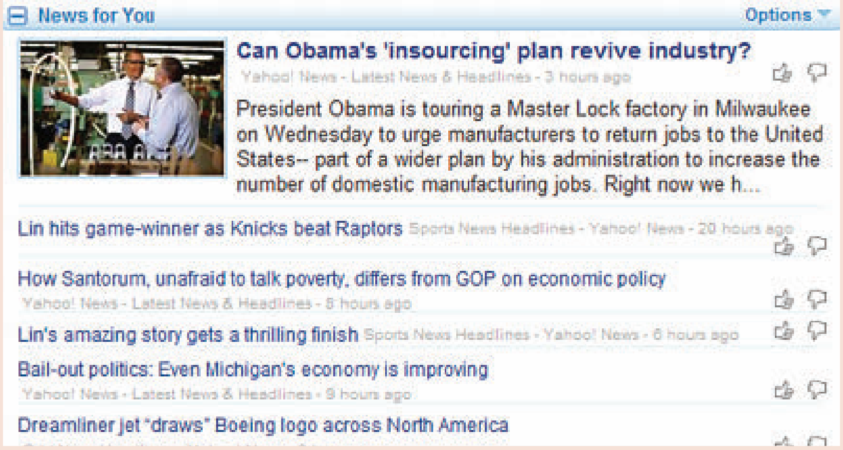


Figure 3. News-For-You module.

Can Obama's 'insourcing' plan revive industry?
Yahoo! News - Latest News & Headlines - 3 hours ago
President Obama is touring a Master Lock factory in Milwaukee on Wednesday to urge manufacturers to return jobs to the United States-- part of a wider plan by his administration to increase the number of domestic manufacturing jobs. Right now we h...

Lin hits game-winner as Knicks beat Raptors Sports News Headlines - Yahoo! News - 20 hours ago

How Santorum, unafraid to talk poverty, differs from GOP on economic policy
Yahoo! News - Latest News & Headlines - 8 hours ago

Lin's amazing story gets a thrilling finish Sports News Headlines - Yahoo! News - 6 hours ago

Bail-out politics: Even Michigan's economy is improving
Yahoo! News - Latest News & Headlines - 9 hours ago

Dreamliner jet "draws" Boeing logo across North America

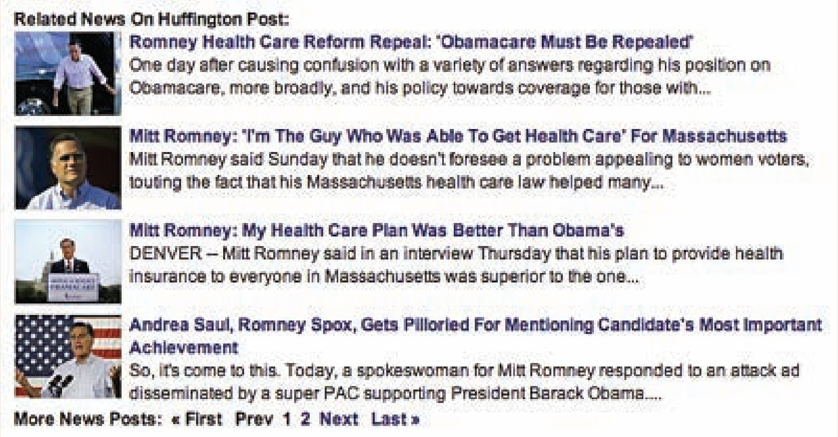


Figure 4. Related-content module on huffingtonpost.com.

Related News On Huffington Post:

Romney Health Care Reform Repeal: 'Obamacare Must Be Repealed'
One day after causing confusion with a variety of answers regarding his position on Obamacare, more broadly, and his policy towards coverage for those with...

Mitt Romney: 'I'm The Guy Who Was Able To Get Health Care' For Massachusetts
Mitt Romney said Sunday that he doesn't foresee a problem appealing to women voters, touting the fact that his Massachusetts health care law helped many...

Mitt Romney: My Health Care Plan Was Better Than Obama's
DENVER – Mitt Romney said in an interview Thursday that his plan to provide health insurance to everyone in Massachusetts was superior to the one...

Andrea Saul, Romney Spox, Gets Pilloried For Mentioning Candidate's Most Important Achievement
So, it's come to this. Today, a spokeswoman for Mitt Romney responded to an attack ad disseminated by a super PAC supporting President Barack Obama....

no use of user or context information. Although simple, this setting poses the challenge of finding the right balance between exploration and exploitation when estimating click-through rates (CTRs) of items using near real-time user click feedback. It also serves as a strong baseline method for non-personalized FMs, especially for portals that employ editors to select or create a small set of high-quality content items. We then extend the application setting to personalized recommendation with a large content pool, which poses a data sparsity challenge because the amount of click feedback at detailed user interest levels is too little to support exploration of even a modest number of items. The key is to reduce dimensionality by leveraging user features and users' past activities on the portal. The techniques reviewed here are also useful for implementing personalized RMs and NMs. After tackling the data sparsity challenge, we discuss multiobjective ranking, which is important for RMs (where semantic relevance to the context page and CTR estimates need to be blended), and more generally, to optimize multiple objectives (for example, clicks, social actions, revenue, time spent on the portal).

Scoring of Items

The fundamental technical challenge in scoring is to estimate the "value" of an item for a given user in a given context. Although one can use semantic relevance between query-document pairs as our score (in content recommendation, a user-context pair is a query), a more appropriate measure is the expected click-through rate or CTR since the main signal from users in content recommendation is whether the user clicks the recommended items. Further, knowing the expected CTR for an item opens the door to a principled approach to ranking items, by weighting the CTR with the utility of a click on that item, which gives the expected utility. Hence, CTR (appropriately weighted) is the primary scoring function we consider in this article. While scores based on other signals like explicit feedback (for example, like/dislike) are useful in some applications and can be estimated by some of the reviewed techniques, they are not our main focus. With known-item CTRs, we could maximize clicks by always recommending the item with highest CTR. But since CTRs are not known, a key task is to accurately estimate the click-through rate (CTR) of each candidate item in the item pool, perhaps through an *exploration* process (displaying each item to some number of user visits). However, exploration has an opportunity cost of not showing items that are empirically better; balancing these two aspects constitutes the explore/exploit trade-off. Dynamic item pools and non-stationarity of CTR over time adds more complexity.

Technical solutions to scoring and ranking in content recommendation modules depend on the goals of the application and the nature of available signals.

The Explore/Exploit Trade-off. To obtain further intuition on the explore/exploit problem, consider a simplified setting with a content pool consisting of two items, where the goal is to obtain an optimal algorithm to maximize overall expected CTR for the next 100 user visits. Note that the solution space here is *astronomical*—there are 2^{100} different possible recommendation sequences (over two trillion!). This is similar in spirit to the classical multi-armed bandit problem, which considers how to dynamically allocate a single resource to alternate projects.[32] Remarkably, an optimal solution exists and involves adaptively changing future decisions based on past feedback as discussed by Gittins.[11] It illustrates the fundamental "explore/exploit" trade-off between "exploit" (display the item that appears to be doing well so far) and "explore" (display the item that may appear inferior, but perhaps due to an unlucky streak, to determine its true popularity). For instance, suppose the estimated CTR after 20 visits for items 1 and 2 are 1/3 and 1/5 respectively. It is tempting to abandon item 2 and persist with item 1 for the remaining 80 visits, but that may not be optimal since the true CTR of item 2 could be *potentially* higher than the estimated one, which is noisy due to the small sample size. The following references offer more details on multi-armed bandit problems.[3,7,11]

For content recommendation, several assumptions required to obtain Gittins' optimal solution are violated. The item pool is not static and changes over time, the CTR themselves could be non-stationary, and the click feedback is delayed (due to delay by users in clicking an item after display and delay in data transmission from Web servers to the backend machines). But perhaps the most difficult issue is the curse of dimensionality introduced due to the need to provide personalized recommendation with large/dynamic content pool, and hence the dearth of experimental budget to estimate popularity at fine resolutions. Hence, content recommendation problems require different solutions and cannot be solved through classical multi-armed bandit schemes.

Here, we discuss technical approaches to solving the explore/exploit problem for content recommendation assuming CTR to be our score function.

The most popular content recommendation. We begin with the problem of recommending the most popular item on a single slot to all users to maximize the total number of clicks. Although simple, this problem of most popular recommendation includes the basic ingredients of content recommendation and also provides a strong baseline for more sophisticated techniques.

Estimating item popularity (CTR) involves several nuances. Ideally, popularity for each item should be estimated by displaying the item to a representative sample of the current user population. For instance, serving most popular at night with popularity estimated using data in the morning may not perform well due to differences in user populations. Several other sources of bias can also affect popularity estimates when using retrospective data collected from existing systems. To protect against such bias, it is useful to update popularity estimates rapidly in an adaptive fashion through procedures like a Kalman filter,[30] using data obtained through randomization, that is, randomly assigning some fraction of visits in a given time epoch to each item in the content pool. The optimal amount of randomization for each item can be computed by using extensions of bandit schemes.[3]

Personalization and large content pools. A natural extension to most popular recommendation is to classify users into coarse segments based on attributes like demographics and geographic location, and then apply most popular recommendation techniques to each segment. Such an approach works well if segments are coarse and item affinities for users within a segment are relatively homogeneous. Techniques like clustering and decision trees[13] can be used to identify such segments.

However, this *segmented most popular recommendation* approach only works when the number of user visits per segment available to explore each candidate item is "sufficiently" large to ensure reliable identification of the highest CTR items. It is therefore challenging to provide personalized recommendations from a large pool of items, since it may not even be possible to show each item to each user even once!

Explore/Exploit with Data Sparsity. Several applications require personalization with a large and dynamic content pool, and this contributes to data sparsity. Approaches to tackling sparsity include:

▸ *Dimensionality reduction.* Creating homogeneous groups through user and item metadata, and/or user behavioral data reduces dimensionality and helps in combating data sparsity.

▸ *Coupling dimension reduction with explore/exploit and online updates.* Although it is ideal to couple explore/exploit techniques with dimension reduction, obtaining an optimal solution is difficult. Heuristics that combine dimension reduction techniques with classical explore/exploit algorithms are often used in practice. In addition, it is important to update model parameters for time-sensitive items in an online fashion using most recent user feedback.

In the following, we first review dimensionality reduction techniques and then discuss how to apply bandit schemes to the resulting low dimensional user and item representations. Subsequently, we review methods for constructing such representations using online updates, and how to effectively initialize these online methods.

Dimensionality reduction techniques. A proper survey of dimensionality reduction goes beyond the scope of this article; we only cover a few representative approaches.

Grouping through hierarchies: In some scenarios, users and/or items are hierarchically organized. For instance, the city in which a user lives is nested within a state, which in turn is nested within a country (Figure 5). If such hierarchies do not exist, hierarchical clustering or decision tree learning[13] may be applied to automatically create hierarchies from data.

Instead of exploring/exploiting individual items for each individual user, one can start with exploring/exploiting coarse content categories for coarse user segments and gradually transition to fine-grained user segments and items as we obtain more data.[17,29]

Dimensionality reduction through linear projections: Another popular approach is to work in a generalized linear model framework.[28] In many ap-

Figure 5. A sample hierarchical organization of a user's location.

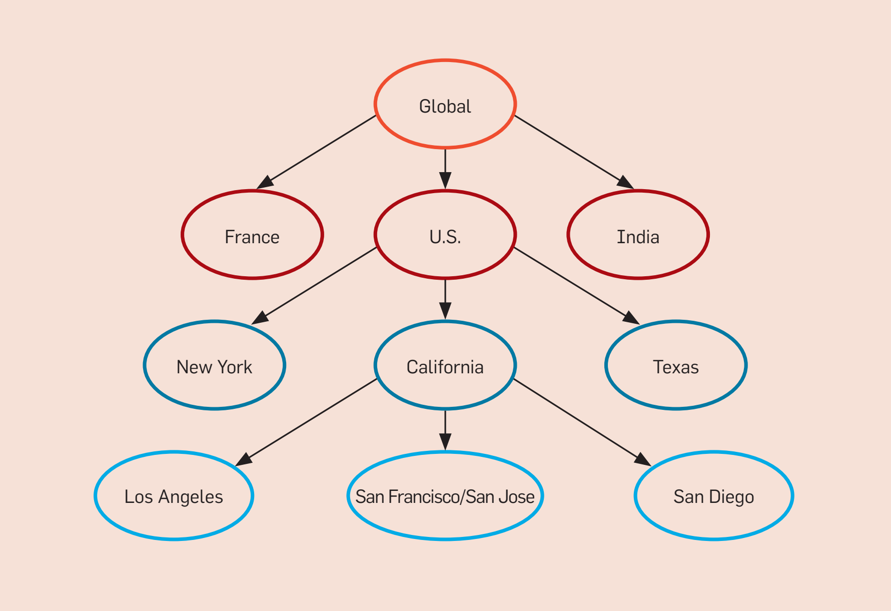
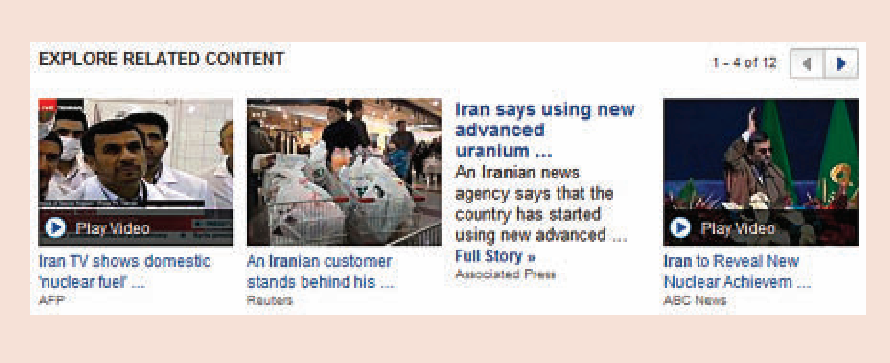


Figure 6. Related content module.



EXPLORE RELATED CONTENT

1 - 4 of 12

Iran TV shows domestic 'nuclear fuel' ...
AFP

An Iranian customer stands behind his ...
Reuters

Iran says using new advanced uranium ...
An Iranian news agency says that the country has started using new advanced ...
Full Story »
Associated Press

Iran to Reveal New Nuclear Achievem ...
ABC News

plications, we have a rich set of user features such as demographics, geolocation, and behavioral activities such as searches. Let us denote by $x_i = (x_{1i}, \ldots, x_{Mi})$ the feature vector for user *i*. The number *M* of such features is typically large (thousands or even millions). A generalized linear model assumes the CTR of item *j* for user *i* is a monotone function of a linear combination of features $x'_i\beta_j$, and the unknown item coefficient vector β_j is estimated by using item interaction data across users. Although such a model reduces the CTR estimation problem from estimating a CTR for each (user, item) pair to estimating *M* coefficients for each item, it is still daunting when *M* is large. One approach is to project β_j into a low-dimensional space through a linear projection matrix *B*; that is, $\beta_j = \beta\theta_j$ where θ_j is low dimensional. The projection *B* can be estimated in an unsupervised fashion by using principal component analysis (PCA)[13] on user features; a supervised approach that uses additional click feedback information provides better performance.[4]

Grouping through collaborative filtering. In typical content recommendation applications, a certain fraction of users tends to interact frequently with the recommendation module. For such users, it is possible to derive features that capture item affinity based on past user interactions alone. It is also possible to derive such features for users who are less frequent by using techniques like collaborative filtering. For instance, based on data from the entire user population, one can estimate associations of the form: "users who like item *A* also like item *B*." The stronger these associations, the smaller the effective dimensionality, because they induce soft constraints on the joint distributions of CTR of all user-item pairs. Such approaches work well in practice even in the absence of other user and item features.[1,8]

Collaborative filtering approaches based on factor models currently provide state-of-the-art performance.[19] The idea is to map user *i* and item *j* into the same Euclidean space as *factor vectors* u_i and v_j respectively—user-item affinity is then given by the inner product u'_iv_j of u_i and v_j, which is a measure of similarity between the two. Unlike explicit interest categories, these groups are latent and are estimated from retrospective data; a small number (a few tens to hundreds) of factors usually provide good performance in applications.

Explore/exploit and dimension reduction. As we discussed earlier, obtaining an optimal explore/exploit solution for personalized recommendation using dimension reduction is non-trivial and hence heuristics are often used in practice. The simplest solution is called ε-greedy. It serves an item selected at random with probability ε to each user visit and, with probability $1 - \varepsilon$, it serves the item having the highest estimated CTR for that user (see Kakade[16] and Langford[20] for more details). Upper-Confidence-Bound (UCB) scheme[7,21] that ranks items based on an overestimate of mean (for example, mean + *k*-std, where *k* is some positive number and std is the estimated standard deviation) and Thompson sampling[37] that sample parameters from the posterior distribution are other commonly used schemes.

Online CTR estimation. CTR estimation models for time-sensitive items needs to be updated frequently using the most recent user feedback. Such online models start with an initial estimate of the model parameters and continuously update parameters as new data becomes available. For instance, consider the factor model that predicts the CTR (on the log-odds scale) of item *j* for user *i* by u'_iv_j. Assume user factor estimates are more stable than items; thus, only v_j needs to be updated in an online manner. From a Bayesian perspective, without any click feedback on item *j*, we postulate a prior distribution for v_j with mean μ_j and some variance. After receiving feedback (click or no-click), we update the prior to obtain the posterior distribution of v_j through Bayes' rule; the resulting posterior serves as the prior for subsequent updates. See Agarwal[5] for an application of this technique to a Web portal.

Initializing online models. For an item *j* that is new or has received little click feedback, CTR prediction depends crucially on the prior mean μ_j, which is the initial estimate. One may naively set μ_j of all items to the same value for all users, say 0. This can be improved by initializing item factors v_j through feature vector z_j for item *j* as $v_j = D(z_j) + \eta_j$, where *D* is a multivariate regression function and η_j are correction terms learning item-specific idiosyncrasies not captured through item features. Similar considerations apply to user factors. We refer the reader to Agarwal[2] and Stern[33] for more details.

Ranking

After obtaining scores of some particular type (for example, CTR estimates) for each user-item pair, one could simply rank items by sorting scores. However, different types of scores (for example, CTR and semantic relevance) may have to be combined, and a number of competing recommendation objectives may have to be balanced. We first present two ranking scenarios to illustrate these nuances, and then discuss how to combine measures for multi-objective ranking.

▸ *Personalized ranking with multiple objectives:* Although CTR is an important signal, other types of complementary signals can also be leveraged to improve personalized ranking. Examples include post-click actions (time spent reading, sharing, commenting, and others), declared interests, explicit feedback (like/dislike), and serendipity measures (to ensure that users are not overly pigeonholed).

▸ *Related-item recommendation:* CTR and semantic relevance must be combined to recommend related items. Figure 6 shows an example of a Related Content Module on a Yahoo! News article page we call the "context article." The candidate items are news articles and videos "related" to the context article. In this setting, contextual information is important. Users expect to see items that are semantically relevant to the article they are currently reading; although highly clicked irrelevant items are not appropriate in this setting, items with low CTRs are also undesired. The semantic relevance measure needs to ensure the candidate item not only is similar to the context item, but also provides sufficient new information not in the context article.[26] Also, CTR estimation needs to take not only the user into consideration, but also the context article, to predict the probability $p(j|i, k)$ that user i would click item j given that he or she likes the context article k, where "like" can be interpreted as click, read, share, and so on. As mentioned earlier, reducing dimensionality is even more important here since data becomes more sparse after incorporating the context. For an example of such dimensionality reduction, see Rendle.[31] To prevent popular items from dominating recommendations, one may down-weight conditional CTR by unconditional CTR (for example, as noted in Davidson[10]), $p(j|i, k)/p(j)$ or $p(j|i, k)/p(j|i)$, or using an affinity measure like the odds ratio.

One potential approach to combining multiple objectives is to compute a score quantifying the value of an item with respect to each of the aspects of interest and then combine these scores through a weighted average. A more general framework of multi-objective optimization can also be used.

Multi-objective optimization. Balancing multiple objectives has broad applicability beyond the two scenarios we just presented. For example, it is often desired to consider various weighted CTR objectives that measure different expected post-click utilities like ad revenue, time spent, likelihood of sharing, and so on. Multi-objective optimization provides an attractive mathematical framework to achieve such trade-offs. Although there is rich and mature literature on multi-objective programming,[34] its application to content optimization is fairly new. For instance, Agarwal et al.[6] use this approach to simultaneously optimize CTR and time spent reading the article to recommend content links on the Yahoo! homepage. Two recent studies[14,35] applied multi-objective optimization in the areas of collaborative filtering.

It is essential to have a rigorous and principled approach to empirically evaluate the quality of solutions.

Evaluation

As discussed earlier, an optimal explore/exploit solution for personalized content recommendation is still elusive. Existing approaches are based on heuristics, so no single strategy necessarily dominates and the actual solutions are application dependent. As such, it is essential to have a rigorous and principled approach to empirically evaluate the quality of solutions.

Strategies to evaluate the performance of an algorithm depend on the aspect we are interested in quantifying. In general, we evaluate two aspects of algorithms used in content recommendation—out-of-sample prediction accuracy and overall recommendation performance.

Evaluating predictive accuracy. A prediction algorithm predicts an unobserved quantity. For example, algorithms that try to predict the CTR of an item by a user fall into this category, including methods discussed earlier in this article. Standard machine-learning or statistical evaluation methods can be used to evaluate the accuracy of predictions. A common approach is to collect a click log from the system that records whether users clicked items shown to them, and then split the log data into a training set and a test set. The training set is used to build a model, which is then evaluated using the test set. Commonly used metrics include log-likelihood of the model on the test data, precision-recall curve, ROC curve, area under ROC curve (AUC).[38] We note that algorithms that predict quantities other than CTR (for example, semantic relevance, time-spent) can also be similarly evaluated as long as ground-truth data is obtainable (for example, by human judgment or processing system logs).

Evaluating recommendation performance. Overall recommendation performance is typically measured by using a notion of "reward" (for example, total number of clicks) that the system receives from its users. Explore/exploit algorithms and ranking algorithms (both of which use the output of some prediction algorithms to decide which items to show to each user) fall into this category. As long as the reward is measurable, algorithms can be compared through online experiments, in which random sub-populations of users are subjected to different serving algorithms to adjust for population bias in measuring performance improvements. Such experiments are frequently conducted by Web portals, and are usually referred to as bucket tests or A/B test. See Kohavi[18] for a survey.

Bucket tests are expensive and may be risky, since an inferior serving algorithm may degrade user experience. It is desired to be able to evaluate an algorithm using retrospective data collected from the system, instead of experimenting with live traffic. Offline evaluation helps in reducing experimental cost and also facilitates research by the broader community, many of whom may not have access to experimental infrastructure. However, it can be difficult to obtain an unbiased estimate of the reward from retrospective data. For example, if an item has never been shown to a user in the retrospective data, estimating the number of clicks that will be generated when the algorithm recommends the item is difficult. If we base the evaluation only on the items that have been shown to the user by the current "serving algorithm," then this evaluation would be biased. Fortunately, as long as the historical algorithm has a non-zero probability of showing each item to each user, the bias can be removed by using importance-sampling techniques. For example, Li et al.[22] developed such an unbiased evaluation method for explore/exploit algorithms, and empirically showed that the reward of an algorithm estimated using retrospective data correlates well with its actual reward in an online bucket test.

Data for research purposes. To facilitate further research in this area, it is important to have benchmark datasets available for evaluating new algorithmic approaches. Several such datasets are available to test predictive accuracy of new approaches,[12,15,40] while not many datasets are available to evaluate the overall recommendation performance. Fortunately, Yahoo![39] provides the first such dataset.

More such datasets are required to facilitate participation by a large number of researchers in this area. This is a challenging task since these datasets are most easily generated by large Web companies but such companies are often constrained by concerns over user privacy. Further collaboration between academic and industrial research is necessary.

Content recommendation techniques have been adopted by every major Web portal to increase user engagement.

Content Recommendations in Current Web Portals

Content recommendation techniques have been adopted by every major Web portal to increase user engagement.[27] We have presented a rigorous technical framework covering all real use cases that we are aware of, rather than a survey of individual methods used by various portals. The actual methods used can be adequately described by a combination of techniques within this framework.

Several portals use methods that are close to related item recommendation techniques described earlier; typically, they use a combination of collaborative filtering and semantic similarity. Such techniques have been successfully used in e-commerce (for example, Amazon[24]) and entertainment (for example, deezer.com). One of the earliest uses of these techniques for content recommendation was by Findory[23] to recommend news articles based on semantic similarity between a user's content profile (obtained through historical reads) and article content, blended with a collaborative filtering component that recommends articles read by a user to others with similar profiles. More recently Das[9] describes an online collaborative filtering framework to recommend news on Google, using pure collaborative filtering (semantic relevance is not incorporated). In a subsequent paper,[25] content-based user profiles are used to enhance the collaborative filtering algorithm and popularity estimates. The YouTube video recommendation system[10] scores a candidate pool of videos in a multi-objective framework with respect to video quality, user specificity and diversity, via a linear combination. The candidate pool of videos for a user is generated by considering related videos (based on co-visitation patterns within the same session) in the relatedness graphs that are at most *n* hops from the seed set (which consists of videos previously watched, commented, shared or liked by the user).

Several portals recommend articles using some estimates of article popularity. For instance, Szabo and Huberman[36] describe methods to estimate long-term article popularity based on historical data. Agarwal et al.[5] showed that this approach suffered from the presence of serving bias in historical data, and used popularity estimation methods based on randomized data to recommend articles on the Yahoo! front page. Several other portals including AOL, LinkedIn and MSN have modules on their home page that rely on methods to estimate article popularity. Factor models to personalize recommendations are known to be used by Netflix[19] for movie recommendations and Yahoo! for personalized content recommendations.[2]

Conclusion

In this article, we described the content recommendation problem and observed that it is sufficiently different from search and ad targeting to merit careful study and distinct technical approaches. We presented a rigorous framework that allows us to capture current recommendation approaches, used widely on several Web portals.

However, challenges and open problems remain, including the following:

▸ *Incorporating externalities:* How does the CTR of an item link depend on other links on the page, and how can we leverage this dependency to better optimize the whole page?

▸ *Combining heterogeneous user feedback from multiple sources:* How can we take advantage of correlations among many kinds of user feedback (for example, clicks, tweets, shares, and email actions) from many sources (for example, browsing and device histories), and combine these signals to better rank items?

▸ *Scalable content curation:* How can we acquire and curate a large number of content items to ensure content quality? Large, high-quality content pools are essential for good personalized recommendations, since we must cater to a wide range of user tastes.

▸ *Algorithm-enabled journalism:* The blending of algorithmic recommendations with journalism and editorial oversight is a fascinating aspect of the rapidly growing role of algorithmic content recommendation. The role of editors and journalists is clearly changing, but algorithms work best when judiciously leveraged with humans in the loop.

References

1. Adomavicius, G. and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowledge and Data Engineering 17* (June 2005), 734–749.
2. Agarwal, D. and Chen, B.-C. Regression-based latent factor models. In *Proceedings of KDD* (2009). ACM Press, New York, 19–28.
3. Agarwal, D., Chen, B.-C. and Elango, P. Explore/exploit schemes for web content optimization. In *Proceedings of ICDM* (2009), 1–10.
4. Agarwal, D., Chen, B.-C. and Elango, P. Fast online learning through offine initialization for time-sensitive recommendation. In *Proceedings of KDD* (2010). ACM Press, New York, 703–712.
5. Agarwal, D., Chen, B.-C., Elango, P., Motgi, N., Park, S.-T., Ramakrishnan, R., Roy, S. and Zachariah, J. Online models for content optimization. In *Proceedings of NIPS* (2008).
6. Agarwal, D., Chen, B.-C., Elango, P. and Wang, X. Click shaping to optimize multiple objectives. In *Proceedings of KDD* (2011). ACM Press, New York, 132–140.
7. Auer, P., Cesa-Bianchi, N. and Fischer, P. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 2002.
8. Das, A., Datar, M., Garg, A. and Rajaram, S. Google news personalization: scalable online collaborative filtering. In *Proceedings of WWW* (2007). ACM Press, New York, 271–280.
9. Das, A., Datar, M., Garg, A, and Rajaram, S. Google news personalization: scalable online collaborative filtering. In *Proceedings of WWW* (2007).
10. Davidson, J., Liebald, B., Liu, J., Nandy, P. Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. and Sampath, D. The youtube video recommendation system. In ACM Conference on Recommender Systems (2010). ACM Press, New York, 293–296.
11. Gittins, J. Bandit processes and dynamic allocation indices. *J. of the Royal Statistical Society B* 41 (1979).
12. GroupLens Research. Movielens Data Sets; http://www.grouplens.org/node/73.
13. Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning.* Springer, 2009.
14. Jambor, T. and J. Wang, J. Optimizing multiple objectives in collaborative ltering. In *Proceedings of the 4th ACM conference on Recommender systems* (2010)., ACM Press, New York, 55–62.
15. Kaggle. http://www.kaggle.com.
16. Kakade, S., Shalev-Shwartz, S. and Tewari, A. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of ICML* (2008). ACM Press, New York, 440–447.
17. Kocsis, L. and Szepesvari, C. Bandit based Monte-Carlo planning. *Machine Learning: ECML, Lecture Notes in Computer Science.* Springer, 2006, 282–293.
18. Kohavi, R., Longbotham, R., Sommerfield, D. and Henne, R. Controlled experiments on the Web: survey and practical guide. *Data Mining and Knowledge Discovery 18* (2009), 140–181. 10.1007/s10618-008-0114-1.
19. Koren, Y., Bell, R. and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer 42*, 8 (2009) 30–37.
20. Langford, J. and Zhang, T. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of NIPS*, (2007).
21. Li, L., Chu, W., Langford, J. and Schapire, R. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM Press, New York (2010) 661–670.
22. Li, L., Chu, W., Langford, J. and Wang, X. Unbiased offine evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining* (2011). ACM Press, New York, 297–306.
23. Linden, G. In http://glinden.blogspot.com/2008/01/brief-history-of-findory.html.
24. Linden, G., Smith, B. and York, J. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing 7*, 1 (2003), 76–80.
25. Liu, J., Dolan, P. and Pedersen, E.R. Personalized news recommendation based on click behavior. In *Proceedings of the 15th International Conference on Intelligent User Interfaces* (2010), 31–40.
26. Lv, Y., Moon, T., Kolari, P., Zheng, Z., Wang, X. and Chang, Y. Learning to model relatedness for news recommendation. In *Proceedings of WWW* (2011). ACM Press, New York, 57–66.
27. Mondaynote. In http://www.mondaynote.com/2009/02/15/recommendation-engines-a-must-for-news-sites, 2009.
28. Nelder, J. and Wedderburn, R. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General) 135* (1972) 370–384.
29. Pandey, S., Agarwal, D., Chakrabarti, D. and Josifovski, V. Bandits for taxonomies: A model-based approach. In *Proceedings of SIAM International Conference on Data Mining* (2007).
30. Pole, A. West, M. and Harrison, P.J. *Applied Bayesian Forecasting & Time Series Analysis.* Chapman-Hall, 1994.
31. Rendle, S. Freudenthaler, C. and Schmidt-Thieme, L. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of WWW* (2010), ACM Press, New York, 811–820.
32. Robbins, H. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc. 58* (1952), 527–535.
33. Stern, D., Herbrich, R. and Graepel, T. Matchbox: Large scale online bayesian recommendations. In *Proceedings of WWW* (2009). ACM Press, New York, 111–120.
34. Steuer, R. *Multi-criteria Optimization: Theory, Computation and Application.* Wiley, 1986.
35. Svore, K.M., Volkovs, M.N. and Burges, C.J. Learning to rank with multiple objective functions. In *Proceedings of the 20th International Conference on World Wide Web* (2011). ACM Press, New York, 367–376.
36. Szabo, G. and Huberman, B.A. Predicting the popularity of online content. *Commun. ACM 53*, 8 (Aug. 2010), 80–88.
37. Thompson, W.R. On the likelihood that one unknown probability exceeps another in view of the evidence of two samples. *Biometrika 25*, 3-4 (1933), 285–294.
38. Witten, I.H., Frank, E. and Hall, M.A. *Data Mining: Practical Machine Learning Tools and Techniques* (3rd Edition). Morgan Kaufmann, Burlington, MA.
39. Yahoo! Academic Relations. R6A–Yahoo! Front Page Today Module User Click Log Dataset, Version 1.0; http://Webscope.sandbox.yahoo.com, 2012.
40. Yahoo! Academic Relations. Yahoo! webscope rating datasets; http://webscope.sandbox.yahoo.com/catalog.php?datatype=r, 2012.

Deepak Agarwal (dagarwal@linkedin.com) is Director of Engineering-Machine Learning at LinkedIn, Mountain View, CA.

Bee-Chung Chen (bchen@linkedin.com) is a senior applied researcher at LinkedIn, Mountain View, CA.

Pradheep Elango (pradheep@fb.com) is senior engineer and technical lead for the Ads Optimization team at Facebook, Menlo Park, CA.

Raghu Ramakrishnan (raghu@microsoft.com) is a Technical Fellow and CTO of information services at Microsoft, Redmond, WA.

interactions
Socialbots:
Voices from the Fronts

interactions
Interactions
with
Big
Data
Analytics

interactions
Media Studies, Mobile Augmented
Reality, and Interaction Design

DOI:10.1145/2461256.2461278

Technical Perspective
Circuit Placement Challenges

By Yao-Wen Chang

FOR A SEMICONDUCTOR circuit with billions of transistors, finding desired locations of circuit components is a challenging task that substantially impacts circuit quality and manufacturing cost. On-chip components must not overlap and should simultaneously optimize several conflicting cost metrics, such as silicon area, circuit performance, and power. Circuit placement—studied since the invention of the integrated circuit (IC)—remains one of the most important steps in VLSI design because it determines the landscape of a silicon chip and increasingly influences other circuit optimizations. Even as a purely computational task, it is difficult. Emerging technology and design challenges have further reshaped the classical placement problem, and thus have provided substantial research opportunities. Consequently, dozens of new placers were developed in the past decade to address the new challenges.

Practical algorithms for VLSI placement trace their roots to three computational approaches: stochastic search based on simulated annealing, repeated min-cut partitioning of hypergraphs, and analytical minimization of specified differentiable functions. Some of these ideas have been successfully combined, but analytical algorithms have recently been recognized to provide the best trade-off between layout quality and scalability for modern VLSI circuits. This trend is not unique to application-specific IC (ASIC) designs, but can also be observed for field-programmable gate array (FPGA) applications. Xilinx Inc., a leading FPGA vendor, recently announced their new FPGA design tool, the Vivado Design Suite, has migrated from the traditional simulated-annealing-based placement to a modern analytical algorithm to improve scalability and design quality. Further, analytical placers have consistently dominated in all recent placement contests organized by IBM Research.

The SimPL placer by Kim et al. is an influential work that furthers the development of analytical placement. The authors formulated a self-contained, concise, and efficient framework to handle modern placement problems. To better understand the contributions of this placer, we start by dissecting the basic flow and structure of analytical placement. Modern analytical placers typically consist of three major steps: Global placement estimates ideal positions for individual circuit components to minimize a predefined cost function (such as interconnect length) while ignoring component overlaps; Legalization removes cell overlaps while trying to preserve the cost; and Detailed placement further improves the legalized placement solution. Global placement is crucial in determining placement quality and speed. Analytical placement models global placement as a mathematical program consisting of an objective function (for example, total interconnect length) and a set of placement constraints (for example, component-density limit), and then minimizes the objective using numerical algorithms. This minimization problem involves four major ingredients: A differentiable function to approximate interconnection length, algorithms to reduce component overlaps (to satisfy optimization constraints), simultaneous optimization of the objective and overlap functions, and the optimization process.

SimPL proposes an elegant integration of the Bound2Bound interconnection length function from the earlier Kraftwerk2 algorithm, geometric partitioning for overlap reduction, region density handling for interconnection length and overlap minimization, and quadratic optimization. SimPL tightly integrates interconnection length and overlap minimization. Like the earlier Vasstu algorithm, SimPL maintains a progression of lower-bound and upper-bound placements that converge to produce a solution. The lower-bound placements are produced by quadratic programming. The upper-bound placements are produced by look-ahead legalization that temporarily spreads out high-density regions to estimate a desirable placement. This concept of look-ahead legalization is similar to what was previously proposed for floorplanning at UCLA and for placement in my group at NTU. However, SimPL develops a new algorithm that implements look-ahead legalization using geometric partitioning and nonlinear scaling. Equally important is the systematic use of look-ahead legalization in global-placement iterations that is characteristic of SimPL. Here the idea is to treat the legalized component locations as fixed anchors (like the Vasstu algorithm) to which the components are tethered with artificial interconnects when the lower-bound placement is produced by quadratic programming. With an appropriate substitution, such connections to fixed locations do not increase the matrix size for quadratic optimization and, in fact, enhance diagonal dominance in matrix solvers, leading to faster convergence.

Among the two major families of analytical placers, quadratic placers (for example, Kraftwerk2 and SimPL) are typically more efficient, while nonlinear (non-quadratic) placers (for example, mPL and NTUplace series) often have better placement quality. To avoid biases with respect to specific circuit structures, more studies based on multiple sets of benchmarks (including at least recently released IBM benchmark suites) would be needed to explore the stability of a placer. Yet, interconnect length and overlap optimization alone is certainly not the end objective of modern placement. Emerging technologies and needs bring up substantial new challenges; some most addressed challenges include large-scale mixed-size placement with millions of cells and hundreds/thousands of big circuit blocks, routability, timing, manufacturability, reliability, power delivery, clock networks, 3D ICs, asynchronous designs, and parallelism. These challenges offer substantial placement research opportunities for the decades to come.

Yao-Wen Chang is a distinguished professor, chair of the Graduate Institute of Electronics Engineering, and Associate Dean of the College of Electrical Engineering and Computer Science, National Taiwan University, Taipei.

DOI:10.1145/2461256.2461279

SimPL: An Algorithm for Placing VLSI Circuits

By Myung-Chul Kim, Dong-Jin Lee, and Igor L. Markov

Abstract

VLSI placement optimizes locations of circuit components so as to reduce interconnect. Formulated in terms of (hyper) graphs, it is NP-hard, and yet must be solved for challenging million-node instances within several hours. We propose an algorithm for large-scale placement that outperforms prior art both in runtime and solution quality on standard benchmarks. The algorithm is more straightforward than existing placers and easier to integrate into timing-closure flows. Our C++ implementation is compact, self-contained and exploits instruction-level and thread-level parallelism. Due to its simplicity and superior performance, the algorithm has been adopted in the industry and was extended by several university groups to multi-objective optimization.

1. INTRODUCTION

The first algorithms for circuit placement have been developed at Bell Labs and IBM Research in the 1960s and followed the divide-and-conquer paradigm. They motivated high-performance heuristics for balanced graph-partitioning by Kernighan and Lin and, later, by Fiduccia and Mattheyses, that minimize edge cut. In the mid-1980s, circuit placement was a key application of the newly invented Simulated Annealing methods. Fifteen years later, the number of components in leading chips grew to the point where annealing was much too slow. The divide-and-conquer framework temporarily regained leadership when it was combined with bottom-up clustering and multi-level partitioning. However, in the 2000s, increasing transistor density again demanded faster algorithms with better performance. Linear programming and network flows were tried with limited success.

Placement optimization gradually became more significant in chip design over the years because the amount of interconnect grows faster than the number of components (except for grid-like circuits such as memory blocks). On-chip interconnect now occupies greater volume than transistors and consumes much power. Additionally, transistor delays improve faster than interconnect delay, which today limits the speed of many chips. This is why circuit placement has recently been integrated with more comprehensive optimizations that can reduce interconnect by restructuring the circuit.[1] But such optimizations need initial component locations that minimize edge lengths. This puts an easy-to-formulate graph problem at the core of sophisticated industrial optimizations. For details the readers are referred to Chapters 4 and 8 of Kahng et al.[12]

Modern techniques for VLSI placement approximate interconnect length by differentiable functions and draw on efficient numerical optimizations. Such *global* placement tolerates various geometric misalignments and small overlaps between rectangular components (represented by graph nodes), which are subsequently repaired by combinatorial algorithms for *legalization* and *detailed* placement. Despite impressive improvements reported by researchers[15] and industry software in the last decade, global-placement algorithms suffer several key shortcomings: (*i*) speed, (*ii*) solution quality, (*iii*) simplicity and integration with other optimizations, and (*iv*) support for multi-threaded execution.

State-of-the-art algorithms for global placement form two families: (*i*) *force-directed* quadratic placers, such as Kraftwerk2,[20] FastPlace3,[22] and RQL,[23] and (*ii*) *nonconvex optimization* techniques, such as APlace2,[8] NTU-Place3,[4] and mPL6.[3] To form an intuition about force-directed algorithms, one thinks of individual interconnects as coil springs subject to Hooke's law and seeks a force-equilibrium (min-energy) configuration. Mathematically, the total interconnect length is captured by a quadratic function of component locations and minimized by solving a large sparse system of linear equations. To discourage component overlap, *forces* are added by pulling components away from high-density areas. These forces are represented by *pseudonodes* and *pseudo-edges*, which extend the original quadratic function.[7] They are updated after each linear-system solve until iterations converge. Nonconvex optimization models interconnect length by more sophisticated differentiable functions that grow linearly with length. These functions are minimized by the nonlinear conjugate gradient method. Component density is modeled by functional terms, which are more accurate than forces, but also requires updates after each change to placement.[4, 8] Algorithms in both categories are used in the industry or closely resemble those in industry placers.

Nonconvex optimization methods previously claimed the best results for academic implementations[4] and industry software, but are significantly slower, which is problematic for modern chip designs with components in many millions. To scale the basic nonconvex optimization framework, best tools in this family employ *hypergraph clustering* and *multi-level/multigrid extensions*, sometimes at the cost of solution quality. Such multilevel placers perform many sequential steps, obstructing efficient parallelization. Moreover, clustering and refinement do not fully benefit from modern multicore CPUs. Owing to their complexity, multilevel placers are also harder to maintain and combine with other optimizations. In particular, clustered circuits obscure analysis

The original version of this paper appeared in the *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (Jan. 2012)

of routing congestion and timing, and complicate circuit restructuring. State-of-the-art force-directed quadratic placers tend to run many times faster than nonconvex optimization, but also use multilevel extensions in their most competitive configurations. Their solution quality is mixed.

In this work, we develop a self-contained technique for global placement based on quadratic programming. It maintains lower-bound and upper-bound placements that converge to a final solution. The upper-bound placement is produced by our new feasibility projection algorithm based on top-down geometric partitioning and nonlinear scaling. Research in VLSI placement includes a fiercely competitive benchmarking component, and we show that our algorithm performs very well on standard benchmarks.

In the remainder of this paper, Section 2 describes the building blocks from which our algorithm was assembled. Section 3 introduces our key ideas and articulates our solution of the *force modulation* problem. The SimPL algorithm is presented in Section 4 along with complexity analysis. Empirical validation is described in Section 5. The use of parallelism is discussed in Section 6.

2. ESSENTIAL CONCEPTS

Circuit placement typically operates on a gate-level netlist, which consists of standard cells (NAND, NOR, MUX, half-adders, etc.) and interconnect. Each standard cell has a rectangular footprint with well-defined area. A cell's output may connect to inputs of multiple other cells—such interconnects are captured by hyperedges, also known as *signal nets*. Given a netlist $\mathcal{N} = (E, V)$ with nets E and nodes (cells) V, *global placement* seeks node locations (x_i, y_i) such that the area of nodes within any rectangular region does not exceed the area of (cell sites in) that region.[a] Some locations of cells may be given initially and fixed. The interconnect objective optimized by global placement is the Half-Perimeter WireLength (HPWL). While easy to calculate, HPWL is a surprisingly good estimate of the length of routed connections. For node locations $\vec{x} = \{x_i\}$ and $\vec{y} = \{y_i\}$, $\text{HPWL}_{\mathcal{N}}(\vec{x}, \vec{y}) = \text{HPWL}_{\mathcal{N}}(\vec{x}) + \text{HPWL}_{\mathcal{N}}(\vec{y})$, where

$$HPWL_{\mathcal{N}}(\vec{x}) = \sum_{e \in E}\left[\max_{i \in e} x_i - \min_{i \in e} x_i\right] \qquad (1)$$

This formula generalizes the so-called Manhattan (taxicab) distance between two points. Given the rigorous public benchmarking infrastructure developed by IBM Research and academic colleagues,[15] consistent improvements by even several percent are considered significant in both academic literature and industry practice. Efficient optimization algorithms approximate $\text{HPWL}_{\mathcal{N}}$ by differentiable functions.

Quadratic optimization. Consider a graph $\mathcal{G} = (E_{\mathcal{G}}, V)$ with edges $E_{\mathcal{G}}$, vertices V, and edge weights $w_{ij} > 0$ for all edges $e_{ij} \in E_{\mathcal{G}}$. The *quadratic objective* $\Phi_{\mathcal{G}}$ is defined as

$$\Phi_{\mathcal{G}}(\vec{x}, \vec{y}) = \sum_{i,j} w_{i,j}\left[(x_i - x_j)^2 + (y_i - y_j)^2\right] \qquad (2)$$

Its x and y components are cast in matrix form[2, 20]

$$\Phi_{\mathcal{G}}(\vec{x}) = \frac{1}{2}\vec{x}^T Q_x \vec{x} + \vec{c}_x^T \vec{x} + \text{const} \qquad (3)$$

The Hessian matrix Q_x captures connections between pairs of movable vertices, while vector $\vec{c}_x$ captures connections between movable and fixed vertices. For more details, the readers are referred to Section 4.3.2 of Kahng et al.[12] When Q_x is nondegenerate, $\Phi_{\mathcal{G}}(\vec{x})$ is a strictly convex function with a unique minimum, which can be found by solving the system of linear equations $Q_x\vec{x} = -\vec{c}_x$. Solutions can be quickly approximated by iterative Krylov-subspace techniques, such as the conjugate gradient (CG) method and its variants.[19] Since Q_x is symmetric positive definite, CG iterations provably minimize the residual norm. The convergence is monotonic,[21] but its rate depends on the spectral properties of Q_x, which can be enhanced by *preconditioning*. In other words, we solve the equivalent system $P^{-1}Q_x = -P^{-1}\vec{c}_x$ for a nondegenerate matrix P, such that P^{-1} is an easy-to-compute approximation of Q_x^{-1}. Given that Q_x is diagonally dominant, we chose P to be its diagonal, also known as the *Jacobi preconditioner*. We deliberately enhance diagonal dominance in Q_x (Section 4.3).

Quadratic placement example. Consider the graph $\mathcal{G}$ and edge weights w_{ij} in Figure 1. Quadratic placement minimizes the separable quadratic cost function $\Phi_{\mathcal{G}}$ in the x and y directions. For the x-direction,

$$\Phi_{\mathcal{G}}(\vec{x}) = w_{12}(x_1 - x_2)^2 + w_{13}(x_1 - x_3)^2 + w_{1f_1}(x_1 - f_1)^2$$
$$+ w_{23}(x_2 - x_3)^2 + w_{34}(x_3 - x_4)^2 + w_{4f_2}(x_4 - f_2)^2$$

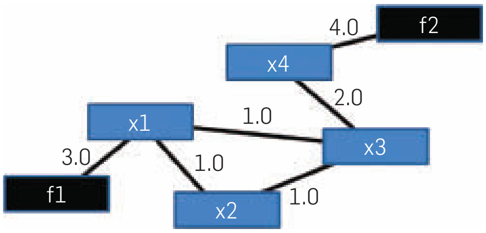
Setting the partial derivatives to 0 (the condition for force equilibrium), we solve for the global minimum cost.

$$\frac{\partial \Phi_{\mathcal{G}}(\vec{x})}{\partial x} = 0 \Leftrightarrow Q_x\vec{x} = -\vec{c}_x \Leftrightarrow \qquad (4)$$

$$\begin{bmatrix} 5.0 & -1.0 & -1.0 & 0.0 \\ -1.0 & 2.0 & -1.0 & 0.0 \\ -1.0 & -1.0 & 4.0 & -2.0 \\ 0.0 & 0.0 & -2.0 & 6.0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 3.0f_1 \\ 0.0 \\ 0.0 \\ 4.0f_2 \end{bmatrix}$$

The connectivity matrix Q_x has entry w_{ij} in the ith row and jth column, and $-\vec{c}_x$ has entry c_i in the ith row. The diagonal entries w_{ii} correspond to the sum of net weights of all connections to movable module i. The off-diagonal entries w_{ij} are calculated as the negative sum of net weights of connections between movable modules i and j, and the resulting connectivity matrix becomes symmetric. Each element c_x for

Figure 1. Blue boxes represent movable modules and black boxes represent fixed modules.



[a] In practice, this constraint is enforced for bins of a regular grid. The layout area is subdivided into equal, disjoint, small rectangles, so as to limit the area of cells placed inside.

a movable module i is calculated as the sum of $w_{ij} \cdot x_j$, where x_j is the pin location of each connected fixed module.With $(f_1, f_2) = (1.0, 3.5)$, a linear system solver finds a unique solution $\bar{x} = [1.4762\ 1.9524\ 2.4286\ 3.1429]^T$ that minimizes the quadratic wirelength $\Phi_{\mathcal{G}}(\bar{x})$.

The Bound2Bound net model.[20] To represent the HPWL objective by the quadratic objective, the netlist $\mathcal{N}$ is transformed into *two* graphs, $\mathcal{G}_x$ and $\mathcal{G}_y$, that preserve the node set V and represent each two-pin net by a single edge with weight 1/*length*. Larger nets are decomposed depending on node locations—for each p-pin net, the *extreme* nodes (min and max) are connected to each other and to each *internal* node by edges, with the following weight

$$w_{x,ij}^{B2B} = \frac{1}{(p-1)|x_i - x_j|} \tag{5}$$

For example, 3-pin nets are decomposed into cliques[14] with edge weight $1/2l$, where l is the length of a given edge. In general, this quadratic objective and the Bound2Bound (B2B) net decomposition capture the HPWL objective exactly, but only for the given placement. As locations change, the approximation error may grow, necessitating multiple updates throughout the placement algorithm.

Most quadratic placers use the placement-independent star or clique decompositions, so as not to rebuild Q_x and Q_y many times.[2, 22, 23] Yet, the B2B model uses fewer edges than cliques ($p > 3$), avoids new variables used in stars, and is more accurate than both stars and cliques.[20]

3. KEY IDEAS IN OUR WORK

Analytic placement techniques first minimize a function of interconnect length, neglecting overlaps between standard cells and macros. This initial step places many cells in densely populated regions, typically around the center of the layout. Cell locations are then gradually spread through a series of placement iterations, during which interconnect length slowly *increases*, converging to a final overlap-free placement (a small amount of overlap is often allowed and later resolved during legalization).

Our algorithm also starts with interconnect minimization, but its next step is unusual—most overlaps are removed using a fast *look-ahead legalizer* based on top-down geometric partitioning and nonlinear scaling. Locations of movable objects in the legalized placement serve as *anchors* that coerce the initial locations to reduce overlap by adding pseudonets to baseline force-directed placement.[7] Each subsequent iteration of our algorithm produces (i) an almost-legal placement that *overestimates* the final result through look-ahead legalization and (ii) an illegal placement that *underestimates* the final result—through linear system solver. The wirelength gap between lower- and upper-bound placements helps monitor convergence (Section 4.3).

Solving the force-modulation problem. A key innovation in SimPL is the interaction between the lower-bound and the upper-bound placements—it ensures convergence to a no-overlap solution while optimizing interconnect length. It solves two well-known challenges in analytic placement: (1) finding directions in which to spread the locations (*force orientation*) and (2) determining the appropriate amount of spreading (*force modulation*).[13, 23] This is unlike previous work, where spreading directions are typically based on *local information*, for example, placers based on nonconvex optimization use *gradient* information and require a large number of expensive iterations. Kraftwerk2[20] orients spreading forces according to solutions of Poisson's equation, providing a global perspective and speeding up convergence. However, this approach does not solve the force-modulation problem, as articulated in Kennings and Vorwerk.[13] The authors of RQL,[23] which can be viewed as an improvement on FastPlace, revisit the force-modulation problem and address it by a somewhat *ad hoc* limit on the magnitude of spreading forces. In our work, *look-ahead legalization algorithm* (Section 4.2), invoked at each iteration, determines both the direction and the magnitude of spreading forces. It is global in nature, accounts for fixed obstacles, and preserves relative placement to ensure interconnect optimization and convergence.

Global placement with look-ahead. The legalized upper-bound placements built at every iteration can be viewed as *look-ahead* because they are used temporarily and not refined directly. The look-ahead placements approximately satisfy constraints (e.g., legality and placement density) while trying to retain quality of current lower-bound placements as much as possible. These locations are then used to update the current lower-bound placements by evolving them toward look-ahead placements. They pull cell locations in lower-bound placements not just away from dense regions but also toward the regions where space is available. Such *area look-ahead* is particularly useful around fixed obstacles, where local information does not offer sufficient guidance. Similar *congestion look-ahead*,[6, 11] *power look-ahead, thermal look-ahead*, and *timing look-ahead* based on legalized placements help integrate our placement algorithm into multi-objective circuit optimizations.

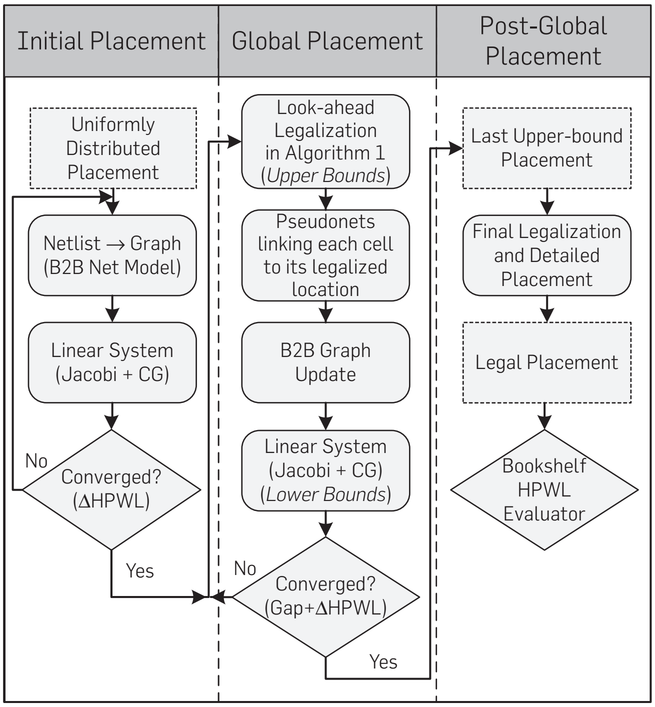
4. OUR GLOBAL PLACEMENT ALGORITHM

Our placement technique consists of three phases: initial placement, global placement iterations, and post-global placement (Figure 2). Initial placement, described next, is mostly an exercise in judicious application of known components. Our main innovation is in the global placement phase. Post-global placement (legalization and detailed placement) is straightforward, given current state of the art.

4.1. Initial placement

Our initial-placement step is conceptually similar to those of other force-directed placers[20, 22, 23]—it entirely ignores cell areas and overlaps, so as to minimize the objective function, a quadratic approximation of total interconnect length. We found that this step notably impacts the final result, as it can determine the overall shape of the final placement solutions. Therefore, unlike FastPlace3[22] and RQL,[23] we use the more accurate B2B net model from Spindler et al.[20] reviewed in Section 2. After the first quadratic solve, we rebuild the circuit graph because the B2B net model is placement-dependent. We then alternate quadratic solves and graph rebuilding until HPWL stops improving. In practice, this requires a small number of

Figure 2. The SimPL algorithm uses placement-dependent B2B net model, which is updated on every iteration. *Gap* refers to the difference between upper and lower bounds.


Initial Placement
Global Placement
Post-Global Placement
Uniformly Distributed Placement
Netlist → Graph (B2B Net Model)
Linear System (Jacobi + CG)
No
Converged? (ΔHPWL)
Yes
Look-ahead Legalization in Algorithm 1 (Upper Bounds)
Pseudonets linking each cell to its legalized location
B2B Graph Update
Linear System (Jacobi + CG) (Lower Bounds)
No
Converged? (Gap+ΔHPWL)
Yes
Last Upper-bound Placement
Final Legalization and Detailed Placement
Legal Placement
Bookshelf HPWL Evaluator

Algorithm 1. Look-ahead Legalization by Top-down Geometric Partitioning and Nonlinear Scaling.

```
Maximum allowed density $\gamma$, where $0 < \gamma < 1$
Placement of cells
Queue of bin clusters $Q = 0$

Identify $\gamma$-overfilled bins and cluster them // Figure 3(a)
foreach cluster $c$ do
  Find a minimal rectangular region $R \supset c$ with density($R$) $\leq \gamma$
  $R.level = 1$
  $Q$.enqueue($R$)
  while !$Q$.empty() do
    $B = Q$.dequeue()
    if (Area($B$) is small enough || $B.level \geq 10$) then
      continue
    $M = \{$movable cells in $B\}$
    $C_c =$ A cutline to evenly split cell area in $M$
    $C_B =$ A cutline to evenly partition whitespace in $B$
    $(S_0, S_1) = \{$two sub-regions of $B$ created by cutline $C_c\}$
    $(M_0, M_1) = \{$movable cells in $S_0, S_1\}$
    $(B_0, B_1) = \{$two sub-regions of $B$ created by cutline $C_B\}$
    Perform NONLINEAR SCALING on $M_0(M_1)$ in $B_0(B_1)$
    $B_0.level = B_1.level = B.level + 1$
    $Q$.enqueue($B_0, B_1$)
  end while
end foreach
```

iterations (5–7), regardless of benchmark size, because the relative ordering of locations stabilizes quickly.

4.2. Look-ahead legalization

Consider a set of cell locations produced by quadratic optimization (lower-bound placement) with a significant amount of overlap as measured using bins of a regular grid. Look-ahead legalization is invoked at each iteration of global-placement process to change the global positioning of those locations, seeking to remove most of the overlap (with respect to the grid) while preserving the relative ordering.[b] This step can be viewed as a projection of the lower-bound placement onto the manifold of feasible placements. The quality of look-ahead legalization is measured by its impact on the entire placement flow. Our look-ahead legalization is based on top-down recursive geometric partitioning and nonlinear scaling (Algorithm 1). Cutlines C_c and C_B are chosen to be vertical at the top level ($R.level = 1$), and they alternate between horizontal and vertical directions with each successive level of top-down geometric partitioning.

Handling density constraints. For each grid bin of a given regular grid, we calculate the total area of contained cells A_c and the total available area of cell sites A_a. A bin is γ-overfilled if its *cell density* A_c/A_a exceeds given density limit $0 < \gamma < 1$. Adjacent γ-overfilled bins are clustered by Breadth-First Search (BFS), and look-ahead legalization is performed on such clusters. For each cluster, we find a minimal containing rectangular region with density $\leq \gamma$ (these regions can also be referred to as "clusters"). A key insight is that overlap removal in a region, which is filled to capacity, is more straightforward because the absence of whitespace leaves less flexibility for interconnect optimization.[c] If relative placement must be preserved, overlap can be reduced by means of *x*- and *y*-sorting with subsequent greedy packing. The next step, *nonlinear scaling*, implements this intuition, but relies on cell-area cutline C_c chosen in Algorithm 1 and shifts it toward the median of available area C_B in the region, so as to equalize densities in the two sub regions (Figure 3).

Nonlinear scaling in one direction is illustrated in Figure 4, where a new region was created by a vertical cutline C_B during top-down geometric partitioning. This region is subdivided into vertical stripes parallel to C_B. First, cutlines are drawn along the boundaries of obstacles present in this region. Each vertical stripe created in this process is further subdivided if its available area exceeds 1/10 of the region's available area. Movable cells in the corresponding subregion created by C_c are then sorted by their distance from C_B and greedily packed into the stripes in that order. In other words, the cell furthest from the cutline is assigned to the furthest stripe. Each subsequent cell is assigned to the furthest unfilled stripe. For each stripe, we calculate the available site area A_a and consider the stripe filled when the area

[b] This formulation is related to the Monge–Kantorovich optimal transport, although in our context runtime is extremely limited and optimal solutions are not required.

[c] In the presence of whitespace, the placer can move cells around without changing their relative ordering. Removing whitespace suppresses this degree of freedom, giving fewer choices to the placer.

Figure 3. Clustering of overfilled bins in Algorithm 1 and adjustment of cell-area to whitespace median by nonlinear scaling (also see Figure 4). Movable cells are shown in blue, obstacles in solid gray.

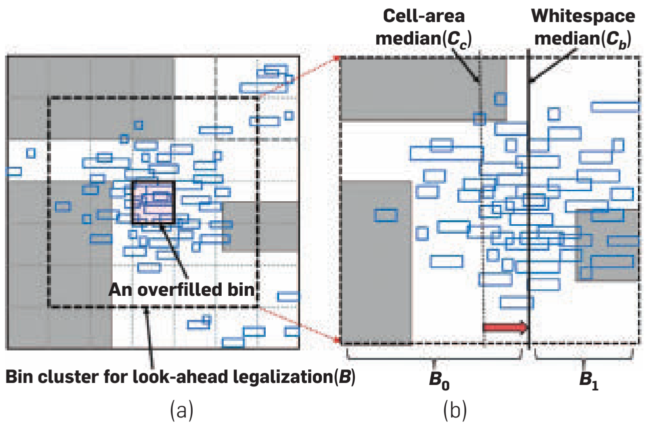
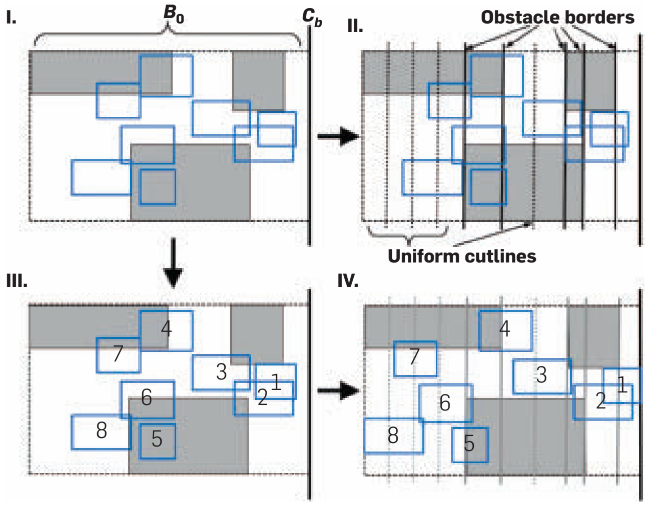


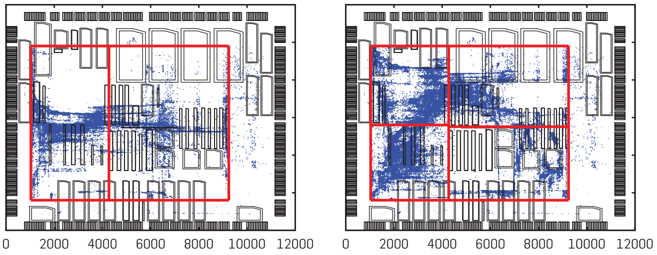
Figure 4. Nonlinear scaling in a region with obstacles (I): the formation of C_B-aligned stripes (II), cell sorting by distance from C_B (III), and greedy cell positioning (IV).



of assigned cells reaches γA_a. Cell locations within each stripe are linearly scaled from current locations (nonlinearity arises from different scaling in different stripes).

Look-ahead legalization applies nonlinear scaling in alternating directions, as illustrated in Figure 5 on one of ISPD 2005 benchmarks. Here, a region R is selected that contains overfilled bins, but is wide enough to facilitate overlap removal. R is first partitioned by a vertical cutline, after which nonlinear scaling is applied in the two new subregions. Subsequently, look-ahead legalization (Algorithm 1) considers each sub region individually and selects different horizontal cutlines. Four rounds of nonlinear scaling follow, spreading cells over the region's expanse (Figure 5).

Figure 5. Nonlinear scaling after the first vertical cut and two subsequent horizontal cuts (ADAPTEC1) between iterations 0 and 1 in Figure 8.



4.3. Global placement iterations

Using legalized locations as anchors. Solving an unconstrained linear system results in a placement with significant amount of overlap. To pull cells away from their initial positions, we gradually perturb the linear system. As explained in Section 4.2, at each iteration of our global placement, top-down geometric partitioning and nonlinear scaling generate a roughly legalized solution. We use these legalized locations as fixed, zero-area anchors connected to their corresponding cells in the lower-bound placement with artificial two-pin pseudonets. Furthermore, following the discussion in Section 2, we note that connections to fixed locations do not increase the size of the Hessian matrix Q, and only contribute to its diagonal elements. For more details, the readers are referred to Section 4.3.2 of. Kahng et al.[12] This enhances diagonal dominance, condition number of $P^{-1}Q$, and the convergence rate of Jacobi-preconditioned CG.

In addition to weights given by the B2B net model on pseudonets, we control cell movement and iteration convergence by multiplying each pseudonet weight by an additional factor $\alpha > 0$ computed as $\alpha = 0.01 \times (1 + \text{Iteration\_Number})$.[d] At early iterations, small α values weaken spreading forces, giving greater significance to interconnect and more freedom to the linear system solver. As the relative ordering of cells stabilizes, increasing α values boost the pull toward the anchors and accelerate the convergence of lower bounds and upper bounds. Mathematically, the α parameter can be viewed as a Lagrange multiplier. The relevant constraint requires that each cell be placed over its anchor, and the (Manhattan) distance between their locations is the penalty for violating the constraint. The α parameter gradually increases and shifts the emphasis of quadratic optimization from reducing interconnect to satisfying constraints (Figure 6).

Convergence criteria similar to that in Section 4.1 can be adopted in global placement. We alternate (1) look-ahead legalization, (2) updates to anchors and the B2B net model, and (3) solution of the linear system, until HPWL of solutions generated by look-ahead legalization stops improving. Unlike in the initial placement step, however, HPWL values of upper-bound solutions oscillate during the first four to seven iterations, as seen in Figure 7. To prevent premature termination, we monitor the gap between the lower and upper bounds. Global placement continues until (1) the gap is reduced to 25% of the gap at the 10th iteration and upper-bound solution stops improving or (2) the gap is smaller

[d] Further improvements in pseudonet weighting and convergence are proposed in Kim and Markov.[9]

Figure 6. An anchor with a pseudonet. The α parameter prices the penalty for the cell being far from its anchor.

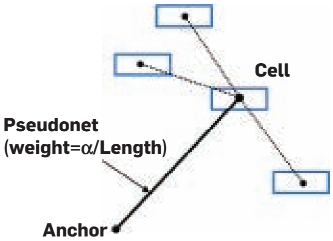
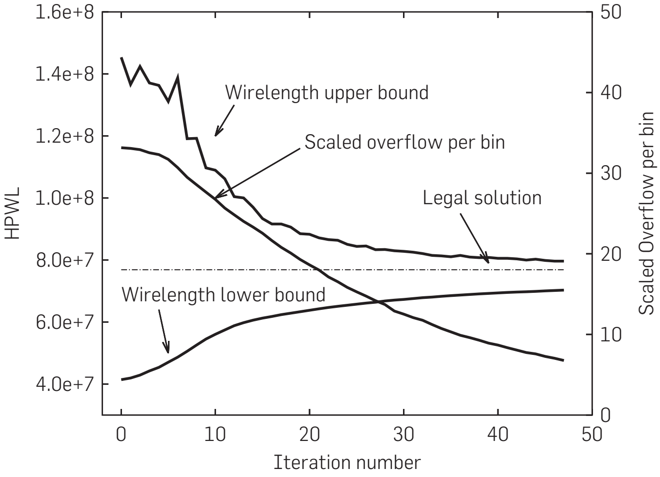


Figure 7. Lower and upper bounds for HPWL, the scaled overflow per bin (a placement density metric) of the lower-bound placement at each iteration, and HPWL of the legal placement (adaptec1).



than 10% of the gap at the 10th iteration. On the ISPD 2005 benchmark suite, only 33–45 iterations are needed. The final set of locations (global placement) is produced by the last look-ahead legalization, as shown in Figure 2.

Convergence is guaranteed by the increasing weights of pseudonets. At each iteration, these pseudonets pull the lower-bound placement toward a legalized upper-bound placement. As the lower-bound placement becomes closer to a legal placement, it exhibits a decreasing amount of cell overlap. This, in turn, results in smaller cell displacements during look-ahead legalization. After the first few iterations, one typically observes monotonic convergence (see Figure 7). A progression of global placement is annotated with HPWL values in Figure 8.

4.4. Asymptotic complexity analysis

The runtime of global placement iterations is dominated by the conjugate gradient (CG) solver and look-ahead legalization. The complexity of each CG invocation is $O\left(m\sqrt{\kappa}\right)$, where κ is the conditioning number of the matrix and m is the number of nonzero elements.[21] The number of nonzeros reflects the number of graph edges in the B2B model of the netlist. It grows linearly with the number of *pins* (cell-to-net connections)—a key size metric of a netlist. Another way to estimate the number of nonzeros is to observe that the average cell degree (the number of nets connected to a cell) is bounded by $d = 5$, or perhaps a slightly larger constant, for practical netlists. Since $m \leq (d + 1)n$ for n cells (including diagonal elements), CG runs in $O\left(n\sqrt{\kappa}\right)$ time.

Asymptotic runtime of look-ahead legalization is dominated by sorting cell locations by their x and y coordinates because nonlinear scaling takes $O(n)$ time (several other linear-time steps take even less time in practice, therefore we do not discuss them). Given that look-ahead legalization operates on blocks of progressively smaller size, we can separately consider its processing pass for the top-level blocks, then the pass for half-sized blocks, etc. Only $O(\log n)$ such passes are required for n cells. Each pass takes $O(n \log n)$ time because top-level blocks do not experience significant overlaps—in fact, each subsequent pass becomes faster because sorting is applied to smaller groups of cells. Hence, look-ahead legalization runs in $O(n \log^2 n)$ time.

We have observed that owing to preconditioning, iteration counts in CG grow no faster than log n, and each iteration takes linear time in n. Therefore, one global placement iteration takes $O(n \log^2 n)$ time. In practice, SimPL requires less than 50 placement iterations, even for million-gate circuits.

5. EMPIRICAL VALIDATION

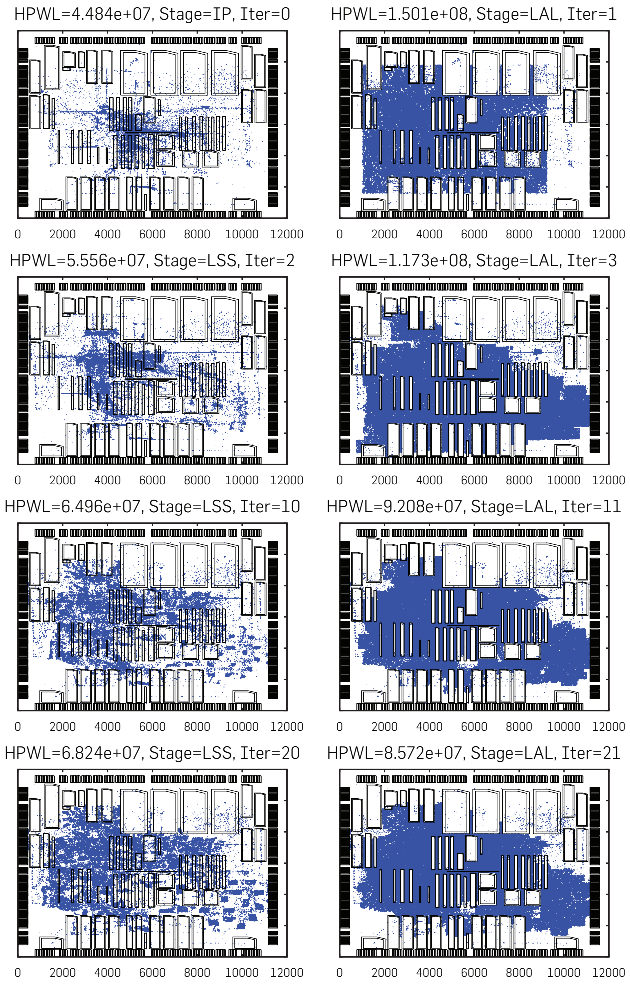
The SimPL global placer is implemented as a stand-alone software package with self-contained I/O, and initial placement and global placement iterations. Living up to its name, it consists of fewer than 5000 lines of C++ code and relies only on standard C++ libraries (shipped with g++ 4.4.0). Single-threaded benchmark runs were performed on an Intel Xeon Quad CPU E31230 (3.2 GHz) Linux workstation with 8GB RAM. We compared SimPL to other academic placers on the ISPD 2005 placement contest benchmark suite[e] with target density $\gamma = 1.0$. Focusing on global placement, we delegate final legalization (into rows and sites) and detailed placement to FastPlace-DP.[16]

Running in a single thread, SimPL completes the entire ISPD 2005 benchmark suite in 1 hour 3 minutes, placing the largest benchmark, BIGBLUE4 (2.18 M cells), in 33 minutes using 2.1GB of memory. We report the runtime breakdown on BIGBLUE4 according to Figure 2, excluding 1.4% runtime for I/O. *Initial placement* takes 5.0% of total runtime, of which 3.7% is spent in CG, and 1.3% in building B2B net models and sparse matrices for CG. *Global placement iterations* take 47.4%, of which 19% is in the CG solver, and 9.9% is in sparse matrix construction and B2B net modeling. Look-ahead legalization takes 17.7%. *Legalization and detailed placement* take 46.2%.

When compared to prior software for VLSI placement (Table 1), SimPL found placements with the lowest interconnect length and was the fastest. On average, SimPL obtains wirelength improvement of 16.26%, 4.12%, 4.23%, and 2.57% versus Capo10.5,[18] NTUPlace3,[4] FastPlace3,[22] and mPL6,[3] respectively. In comparison, one step of Moore scaling reduces interconnect by 30% at the cost several billion dollars. SimPL was 7.28 times faster than mPL6,

[e] http://archive.sigda.org/ispd2005/contest.htm.

Figure 8. A progression of global placement snapshots from different iterations and algorithm steps (adaptec1). IP = Initial Placement, LAL = Look-ahead Legalization, LSS = Linear System Solver. Left-side placements show lower bounds and right-side placements show upper bounds.


HPWL=4.484e+07, Stage=IP, Iter=0

HPWL=1.501e+08, Stage=LAL, Iter=1

HPWL=5.556e+07, Stage=LSS, Iter=2

HPWL=1.173e+08, Stage=LAL, Iter=3

HPWL=6.496e+07, Stage=LSS, Iter=10

HPWL=9.208e+07, Stage=LAL, Iter=11

HPWL=6.824e+07, Stage=LSS, Iter=20

HPWL=8.572e+07, Stage=LAL, Iter=21

which appears to be the strongest preexisting placer. SimPL was 1.12 times faster than FastPlace3—previously the fastest academic software. Multi-objective placers based on SimPL[6, 11] also demonstrate their consistent speed advantages over other state-of-the-art placers, and especially so on larger circuit instances.

6. EXPLOITING PARALLELISM

Further speed-up is possible for SimPL on workstations with multicore CPUs.

Algorithmic details. Runtime bottlenecks in the sequential variant of the SimPL algorithm (Section 5)—updates to the B2B net model and the CG solver—can be parallelized. Given that the B2B net model is separable, we process the x and y cases in parallel. When more than two cores are available, we split the nets of the netlist into equal groups that can be processed by multiple threads. To parallelize the CG solver, we applied a coarse-grain *row partitioning* scheme to the Hessian Matrix Q, where different blocks of rows are assigned to different threads using OpenMP.[5] A core operation in CG is the Sparse Matrix-Vector multiply (SpMxV). Memory bandwidth is a known bottleneck and becomes more critical when multiple cores access the main memory through a common bus. We reduce memory bandwidth demand of SpMxV by using the *CSR* (Compressed Sparse Row)[19] memory layout for the Hessian matrix Q.

Our implementation exploits streaming SIMD extensions level 2 (SSE2)[17] that perform several floating-point operations at once, especially in the conjugate gradient solver. In practice, the impact of parallelization depends on the relation between CPU speed and memory bandwidth.

```
// inner product of two float vectors x and y
float inner_prod(vector<float> &x, vector<float> &y)
{
 __m128 thread_acc[NUM_THREADS], X, Y;
 float temp[4], inner_product=0.0;
 int i;
 for(int j = 0; j < NUM_THREADS; j++)
   thread_acc[j]=_mm_setzero_ps();
 #pragma omp parallel for private(X,Y) lastprivate(i)
 ...
     schedule(static) ordered num_threads(NUM_THREADS)
 for (i=0; i <= x.size()-4; i+=4)
 {
   X = _mm_load_ps(&x[i]);
   Y = _mm_load_ps(&y[i]);
   thread_acc[omp_get_thread_num()] = ...
     _mm_add_ps(thread_acc[omp_get_thread_num()], ...
     _mm_mul_ps(X,Y));
 }
 for(int j = 1; j < NUM_THREADS; j++)
   thread_acc[0]=_mm_add_ps(thread_acc[0],thread_
   acc[j]);
 _mm_store_ps(temp, thread_acc[0]);
 inner_product = temp[0] + temp[1] + temp[2] + temp[3];
 for ( ; i < x.size(); i++)
   inner_product += x[i] * y[i];
 return inner_product;
}
```

Listing 1. Sample code for OpenMP and SSE2 parallelization for the inner-product operation.

After we parallelized the CG solver, look-ahead legalization became a bottleneck and needed to be parallelized as well. To this end, top-down partitioning generates an increasing number of sub-tasks of similar sizes which can be solved independently. Let Q_g be the global queue of bin cluster from Algorithm 1 and Q_i be the private queue of bin clusters of thread i. First, we statically assign initial bin clusters to available threads such that each thread has similar number of bin clusters to start. After each level of top-down geometric partitioning and nonlinear scaling in such a bin cluster, each thread generates two sub-clusters with similar numbers of cells. Then, thread t_i adds only one of two sub-clusters to its private queue Q_i for the next level of top-down geometric partitioning and nonlinear scaling, while the remainder is added to Q_g. Whenever Q_i becomes empty, the thread t_i dynamically retrieves clusters from Q_g. The number of clusters N to be retrieved is given by $N = \max(Q_g.size()\,/\,NUM\_THREADS, 1)$

Table 1. Comparison of HPWL (×10e6) and runtime (minutes) on ISPD 2005 benchmarks. Each placer ran as a single thread on a 3.2 GHz Intel CPU. FastPlace-DP took 40% of runtime for SimPL and FastPlace.

Benchmark			Capo10.5		NTUPlace3		FastPlace3.0		mPL6		SimPL	
Name	#Cells	#Nets	HPWL	Time	HPWL	Time	HPWL	Time	HPWL	Time	HPWL	Time
ADAPTEC1	211 K	221 K	88.14	21.08	81.82	7.62	78.67	2.03	77.93	15.65	*77.42*	*2.01*
ADAPTEC2	255 K	266 K	100.25	25.44	*88.79*	7.07	94.06	2.88	92.04	16.20	91.01	*2.60*
ADAPTEC3	452 K	467 K	276.80	62.19	214.83	14.33	214.13	6.51	214.16	48.29	*203.84*	*5.44*
ADAPTEC4	496 K	516 K	231.30	64.60	195.93	14.55	197.50	6.11	193.89	45.90	*184.70*	*4.88*
BIGBLUE1	278 K	284 K	110.92	33.42	98.41	12.32	96.65	*3.09*	96.80	20.43	*94.66*	3.88
BIGBLUE2	558 K	577 K	162.81	64.44	151.55	23.25	155.75	6.11	152.34	54.02	*145.87*	*5.01*
BIGBLUE3	1.10 M	1.12 M	405.40	146.35	360.66	44.90	365.16	18.87	*344.10*	75.75	351.55	*15.51*
BIGBLUE4	2.18 M	2.23 M	1016.19	453.72	866.43	100.48	836.20	28.83	829.44	163.15	*790.28*	*23.26*
Geometric mean			*1.19×*	*11.55×*	*1.04×*	*3.32×*	*1.04×*	*1.12×*	*1.03×*	*7.28×*	*1.00×*	*1.00×*

Figure 9. Speedup for conjugate gradient (CG) with SSE instructions, B2B net model construction (B2B), and top-down geometric partitioning and nonlinear scaling (T&N).

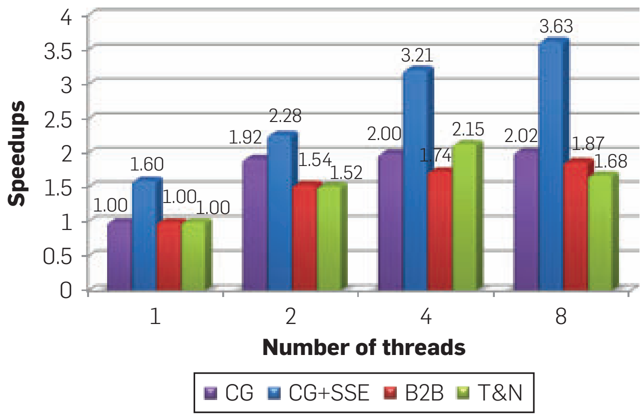
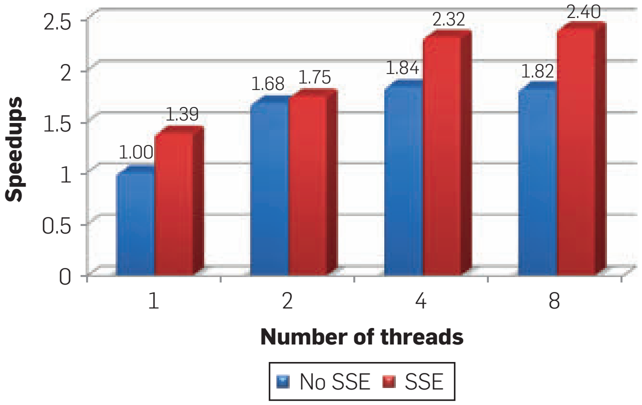


Figure 10. Speedup ratios for global placement on the ISPD 2005 benchmark suite.



Empirical studies evaluated SimPL on an 8-core AMD-based system with four dual-core CPUs and 16GB RAM. Each CPU was Opteron 880 processor running at 2.4 GHz with 1024KB cache. Single-thread execution is compared to eight-thread execution in Figure 9. Our combination of multi-threading and SIMD instruction-level parallelization was 1.6 times faster on average than parallelization based on multi-threading alone. Theoretically, using SIMD instruction-level parallelization may speed up CG by at most four times. However, SIMD-based implementation of SpMxV only provided marginal speedups. This is because irregular memory access patterns of SpMxV prohibit the aligned loading of values (*MOVAPS* or *_mm_load_ps* in Listing 1) to SSE registers. Nevertheless, SSE instructions were helpful elsewhere and contributed to the overall speedup in global placement. The overall speedups in global placement runtimes are shown in Figure 10. Solution quality did not appreciably change, but peak memory usage increased by 1.91 times whereas global placement was 2.4 times faster. The speedups saturate for more than four threads as look-ahead legalization scales poorly. The initial placement stage was accelerated by about three times. While CG remained the runtime bottleneck of SimPL on eight threads (36% of global placement), look-ahead legalization became a close second (>31% of global placement).

7. CONCLUSION

In this work, we developed an algorithm for large-scale VLSI placement. Typical state-of-the-art placers require over 100,000 lines of C++ code, but our self-contained implementation of SimPL uses fewer than 5000 lines.[f] The algorithm is iterative and maintains two placements—one computes a lower bound and one computes an upper bound on the final wirelength. These two placements interact, ensuring stability and fast convergence of the algorithm. The upper-bound placement is produced by a new *feasibility projection* algorithm—*look-ahead legalization*.

The SimPL algorithm has seen rapid adoption since its publication at ICCAD 2010. Two placers[6, 11] based on the SimPL framework finished in top three at the ISPD 2011, DAC 2012, and ICCAD 2012 routability-driven placement contests organized by IBM Research. In particular, He et al.[6] successfully

[f] The SimPL binary is available upon request.

reimplemented SimPL without having access to our source code. Recent industry and academic software packages for ASIC and FPGA placement are now using similar algorithms. The SimPL algorithm has been extended to multilevel optimization within an industry infrastructure, which currently produces the best HPWL results on average.[10] In this context, SimPL's reduced complexity enables fast implementation, parallel processing, and effective software maintenance. Upper-bound placements help integrate timing and congestion optimizations; the baseline SimPL algorithm has been extended to multi-objective optimization, including power-/thermal-/structure-aware placement as summarized and referenced in Kim and Markov.[9] The recent prosperity of SimPL-derived algorithms suggests the applicability of *look-ahead* techniques to other constrained-optimization problems.

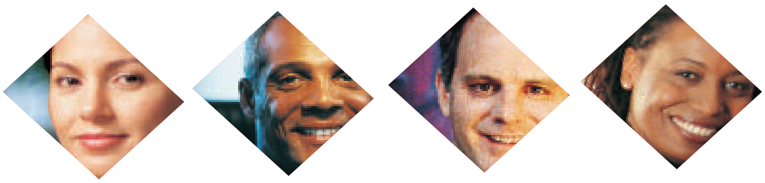
Acknowledgment

This work was supported by Texas Instruments and the Semiconductor Research Corporation (SRC) Task 2264.001 funded by Intel and IBM. C

References

1. Alpert, C.J., et al. Techniques for fast physical synthesis. *Proc. IEEE 95*, 3 (2007), 573–599.
2. Brenner, U., Struzyna, M., Vygen, J. BonnPlace: Placement of leading-edge chips by advanced combinatorial algorithms. *IEEE TCAD 27*, 9 (2008), 1607–1620.
3. Chan, T.F., et al. mPL6: Enhanced multilevel mixed-size placement. *ISPD* (2006), 212–214.
4. Chen, T.C., et al. NTUPlace3: An analytical placer for large-scale mixed-size designs with preplaced blocks and density constraints. *IEEE TCAD 27*, 7 (2008), 1228–1240.
5. Dagum, L., Menon, R. OpenMP: An industry standard API for shared-memory programming. *IEEE Comput. Sci. Eng.* (1998), 46–55.
6. He, X., Huang, T., Xiao, L., Tian, H., Cui, G., Young, E.F.Y. Ripple: An effective routability-driven placer by iterative cell movement. *ICCAD* (2011), 74–79.
7. Hu, B., Marek-Sadowska, M. FAR: Fixed-points addition & relaxation based placement. *ISPD* (2005), 161–166.
8. Kahng, A.B., Wang, Q. A faster implementation of APlace. *ISPD* (2006), 218–220.
9. Kim, M.C., Markov, I.L. ComPLx: A competitive primal-dual Lagrange optimization for global placement. *DAC* (2012), 747–752.
10. Kim, M.C., et al. MAPLE: Multilevel adaptive PLacEment for mixed-size designs. *Proc. ISPD* (2012).
11. Kim, M.C., Hu, J., Lee, D., Markov, I.L. A SimPLR method for routability-driven placement. *ICCAD* (2011), 67–73.
12. Kahng, A.B., Lienig, J., Markov, I.L., Hu, J. *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer, 2011, 312.
13. Kennings, A.A., Vorwerk, K. Force-directed methods for generic placement. *IEEE TCAD 25*, 10 (2006), 2076–2087.
14. Kleinhans, J.J., et al. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE TCAD 10*, 3 (1991), 356–365.
15. Nam, G.J., Cong, J. Modern Circuit Placement: Best Practices and Results, Springer, 2007.
16. Pan, M., Viswanathan, N., Chu, C. An efficient & effective detailed placement algorithm. *ICCAD* (2005), 48–55.
17. Raman, S.K., Pentkovski, V., Keshava, J. Implementing streaming SIMD extensions on the Pentium III processor. *IEEE Micro 20*, 4 (2000), 47–57.
18. Roy, J.A., et al. Capo: Robust and scalable open-source min-cut floorplacer. *ISPD* (2005), 224–226.
19. Saad, Y. Iterative methods for sparse linear systems. *SIAM* (2003).
20. Spindler, P., Schlichtmann, U., Johannes, F.M. Kraftwerk2 – A fast force-directed quadratic placement approach using an accurate net model. *IEEE TCAD 27*, 8 (2008), 1398–1411.
21. Trefethen, L.N., Bau, D. Numerical linear algebra. *SIAM* (1997), 296–298.
22. Viswanathan, N., Pan, M., Chu, C. FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. *ASPDAC* (2007), 135–140.
23. Viswanathan, N., et al. RQL: Global placement via relaxed quadratic spreading and linearization. *DAC* (2007), 453–458.

Myung-Chul Kim, Dong-Jin Lee, and Igor L. Markov ([mckima, ejdjsy, markov]@eecs.umich.edu), Department of EECS, University of Michigan, Ann Arbor.



CAREERS

Dalhousie University

Limited Term Assistant Professor position in the Faculty of Computer Science

Assistant Professor

Dalhousie University (http://www.dal.ca) invites applications for a three year limited term position at the **Assistant Professor** level in the Faculty of Computer Science (http://www.cs.dal.ca) that currently has 30 faculty members, approximately 425 undergraduate majors and 240 master's and doctoral students. The Faculty partners with other Faculties in the University to offer the Master of Electronic Commerce, Master of Health Informatics and Master of Science in Bioinformatics programs, and is an active participant in the Interdisciplinary PhD program.

Dalhousie University is located in Halifax, Nova Scotia (http://www.halifaxinfo.com/), which is the largest city in Atlantic Canada and affords its residents outstanding quality of life.

The Faculty welcomes applications from outstanding candidates in Computer Science. An applicant should have a PhD in Computer Science or related area and be comfortable teaching core computer science courses, particularly Software Engineering. Evidence of a strong commitment to and aptitude for research and teaching is essential. The ideal candidate will be open to collaborative research within the faculty and add to or complement existing research strengths and strategic research directions of the Faculty.

Applications should include an application letter, curriculum vitae, a statement of research and teaching interests, sample publications, and the names, email addresses and physical addresses of three referees. The application must include the Equity Self-Identification form (see the URL below). All documents are to be submitted to the email address below as PDF files.

Applicants should provide their referees with the URL of this advertisement (see below), and request that they forward letters of reference by email to the same address.

Applications will be accepted until June 30, 2013. This position may be subject to budgetary constraints.

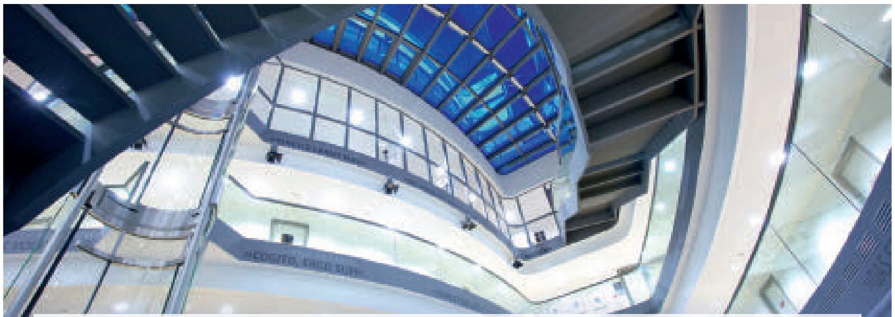
All qualified candidates are encouraged to apply; however Canadian and permanent residents will be given priority. Dalhousie University is an Employment Equity/Affirmative Action Employer. The University encourages applications from qualified Aboriginal people, persons with a disability, racially visible persons and women.

Submission Address for application documents and reference letters: appointments@cs.dal.ca

Location of this advertisement: www.cs.dal.ca/PTTAdvertisement

Self-Identification form (PDF): http://hrehp.dal.ca/Files/Academic_Hiring_%28For/selfid02.pdf

Self-Identification form (Word): http://hrehp.dal.ca/Files/Academic_Hiring_%28For/selfid02.doc



The **Hasso Plattner Institute** for Software Systems Engineering GmbH (HPI) in Potsdam is Germany's university excellence center in Computer Science. Since 2012, the Institute also offers interactive online courses via its MOOC-platform **openhpi.de** that are open to everyone.

Annually, the Institute's Research School seeks talented junior researchers and accordingly offers

8 Ph.D. Scholarships and 2 Postdoc Scholarships

The **HPI Research School** focuses on the foundation and application of large-scale, highly complex and interconnected IT systems. With its interdisciplinary and international structure, the Research School interconnects the HPI research groups as well as its international branches at Cape Town University, Technion - Israel Institute of Technology and Nanjing University. The **HPI Future SOC Lab**, a state-of-the-art computer center, enriches the academic work at the HPI Research School.

The HPI professors and their research groups ensure high quality research and will supervise Ph.D. students in the following topic areas: Human Computer Interaction, Prof. Dr. Patrick Baudisch; Computer Graphics Systems, Prof. Dr. Jürgen Döllner; System Engineering and Modeling, Prof. Dr. Holger Giese; Software Architecture, Prof. Dr. Robert Hirschfeld; Internet Technologies and Systems, Prof. Dr. Christoph Meinel; Information Systems, Prof. Dr. Felix Naumann; Enterprise Platform and Integration Concepts, Prof. Dr. h.c. Hasso Plattner; Operating Systems and Middleware, Prof. Dr. Andreas Polze; Business Process Technology, Prof. Dr. Mathias Weske

If you have prior experience in any of these areas, you are invited to submit a full application with the following documents: curriculum vitae and copies of certificates/transcripts, brief research proposal, work samples/copies of relevant scientific work (e.g. master's thesis), and a letter of recommendation.

Applications must be submitted by August 15th of the respective year.
Positions are usually available at the beginning of October. Please send your applications to: **research-school-application@hpi.uni-potsdam.de**

For more information on HPI and its HPI Research School see:
http://www.hpi.uni-potsdam.de/research_school

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Pontificia Universidad Catolica de Chile

Full time MIS Faculty

Full-time faculty position in Information Systems Management and closely related disciplines at top Chilean University: Pontificia Universidad Catolica de Chile. Deadline 7/31. More information at: www.ing.puc.cl/information_systems

Princeton University

Computer Science Department

Princeton University

The Department of Computer Science seeks applications from outstanding teachers to assist the faculty in teaching our introductory course sequence or some of our upper-level courses.

Depending on the qualifications and interests of the applicant, job responsibilities will include such activities as teaching recitation sections and supervising graduate-student teaching assistants; grading problem sets and programming assignments; supervising students in the grading of problem sets and programming assignments; developing and maintaining online curricular material, classroom demonstrations, and laboratory exercises; and supervising undergraduate research projects. An advanced degree in computer science, or related field, is required (PhD preferred).

The position is renewable for 1-year terms, up to six years, depending upon departmental need and satisfactory performance.

To apply, please submit a cover letter, CV, and contact information for three references to https://jobs.cs.princeton.edu/lecturer/

Requisition # 1200313

Princeton University is an equal opportunity employer and complies with applicable EEO and affirmative action regulations.

WhereScape USA

Principal Managing Software Architect

PRINCIPAL MANAGING SOFTWARE ARCHITECT for WhereScape USA, Inc. in Portland, Oregon. Duties: oversee work of software solution architects and occasionally act as senior solutions architect to: provide data warehousing technical support for field sales & pre-sales technical personnel involved in demand-creation marketing activities, customer-specific sales campaigns, proof-of-content exercises, product installation & after-sale customer support; manage progress and development of our solutions architects, including performance reviews, internal training and development of training materials;

Faculty Search

ShanghaiTech University

School of Information Science and Technology

The School of Information Science and Technology (SIST) in the new ShanghaiTech University invites applications to fill multiple tenure-track and tenured positions. Candidates should have an exceptional academic record or strong potential in frontier areas of information sciences.

ShanghaiTech is founded as a world-class research university for training future scientists, entrepreneurs, and technology leaders. Besides keeping a strong research profile, successful candidates should also contribute to undergraduate and graduate education within SIST.

Compensation and Benefits:
Salary and startup fund are highly competitive, commensurate with academic experience and accomplishment. ShanghaiTech also offers a comprehensive benefit package which includes housing. All regular faculty members are hired within ShanghaiTech's new tenure-track system commensurate with international practice and standards.

Academic Disciplines:
We seek candidates in all cutting edge areas of information science and technology. Our recruitment focus includes, but is not limited to: computer architecture and technologies, nano-scale electronics, high speed and RF circuits, intelligent and integrated signal processing systems, computational foundations, big data, data mining, visualization, computer vision, bio-computing, smart energy/power devices and systems, next-generation networking, as well as inter-disciplinary areas involving information science and technology.

Qualifications:
- Well developed research plans and demonstrated record/potentials;
- Ph.D. (Electrical Engineering, Computer Engineering, Computer Science, or related field);
- A minimum relevant research experience of 4 years.

Applications:
Submit (all in English) *a cover letter, a 2-page research plan, a CV including copies of 3 most significant publications, and names of three referees* to: **sist@shanghaitech.edu.cn**.

Deadline: September 15th, 2013 (or until positions are filled). We have 10 positions for this round of faculty recruitment.

For more information, visit http://www.shanghaitech.edu.cn.

UNIVERSITY OF KENTUCKY

Director
Center for Visualization & Virtual Environments

The University of Kentucky's Center for Visualization & Virtual Environments invites applications from outstanding individuals to serve as Director of the Center. Candidates having an exceptional record of research productivity and who have a demonstrated ability to lead an interdisciplinary unit are encouraged to apply. External applicants will be considered for a tenured faculty position at the associate to full professor level, with the departmental affiliation dependent on the candidate.

Candidates with expertise in any area related to visualization, including but not limited to machine vision, medical imaging, and pattern recognition, will be considered. Successful candidates will have: 1) a clear vision for advancing the Center; 2) experience leading large collaborative proposals; 3) experience managing large and complex budgets; 4) experience with supervising staff; 5) a willingness to work with a wide range of internal and external constituencies; and 6) a commitment to advancing diversity among faculty, staff, and students.

UK offers strong collegial collaboration, with on campus proximity to the Colleges of Architecture, Arts & Sciences, Business and Economics, Dentistry, Education, Engineering, Fine Arts, Medicine, and Pharmacy, and to the UK Hospital. A competitive salary and start-up package will be available, as well as access to excellent core research facilities.

Review of applications will begin immediately and will continue until the position is filled. Applicants should apply for position SM545908 on-line at http://ukjobs.uky.edu. Submit PDF files consisting of a letter of interest, complete curriculum vitae, statement of research goals, vision statement for the Center, and contact information for three professional references.

Please visit http://vis.uky.edu for more information about the Center.

The University of Kentucky is an Affirmative Action/Equal Opportunity Employer. Women and minorities are encouraged to apply.

VISCENTER

represent us as our Chief Technical Officer at public forums including trade shows, conferences & other industry & public gatherings. (WhereScape USA, Inc. is the sales and service arm of WhereScape Ltd., a New Zealand software development company & manufacturer of WhereScape RED, a unique, proprietary data warehouse life cycle management software product.) 60-plus hrs/wk; hours vary. Frequent short-term U.S. & foreign travel to meet with parent company, attend trade shows & other conferences & meet with clients onsite (60 days or less at any single location). Requirements: Bachelor Degree in Electrical Engineering or Computer Science and five years of experience as systems analyst or senior solutions software architect. Experience must include work with: A) WhereScape RED or at least 2 data warehouse lifecycle automation tools such as Kalido, BIReady or Oracle Data Integrator; B) implementing data warehouse & data mart schema using dimensional modeling techniques & business intelligence software such as Tableau, Microstrategy, Business Objects, Cognos, QlikTech or Spotfire against dimensional data marts & warehouses; C) data cubing software such as TBIO from Teradata and MicroSoft SQL Server Analysis Services and design & administrative levels of SQLServer, Netezza, Oracle, IBM DB2 and Teradata; D) WhereScape 3D software or software such as ER/Win, Embarcadero, Oracle Designer or Sybase PowerDesigner; E) a distributed sales & service organization, as well as with an offshore Research & Development (R&D) organization; F) XP, Agile or other non-traditional project management & execution software; and G) developing & documenting business & technical requirements in communications media & delivering those requirements to end-user IT & business teams to gain consensus, buy-in & project approval. Experience may be concurrent. Must have proof of legal authority to work in the United States. Apply by email to marc@wherescape.com with resume, salary requirements, reference list and cover letter listing your qualification for each requirement.

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.**

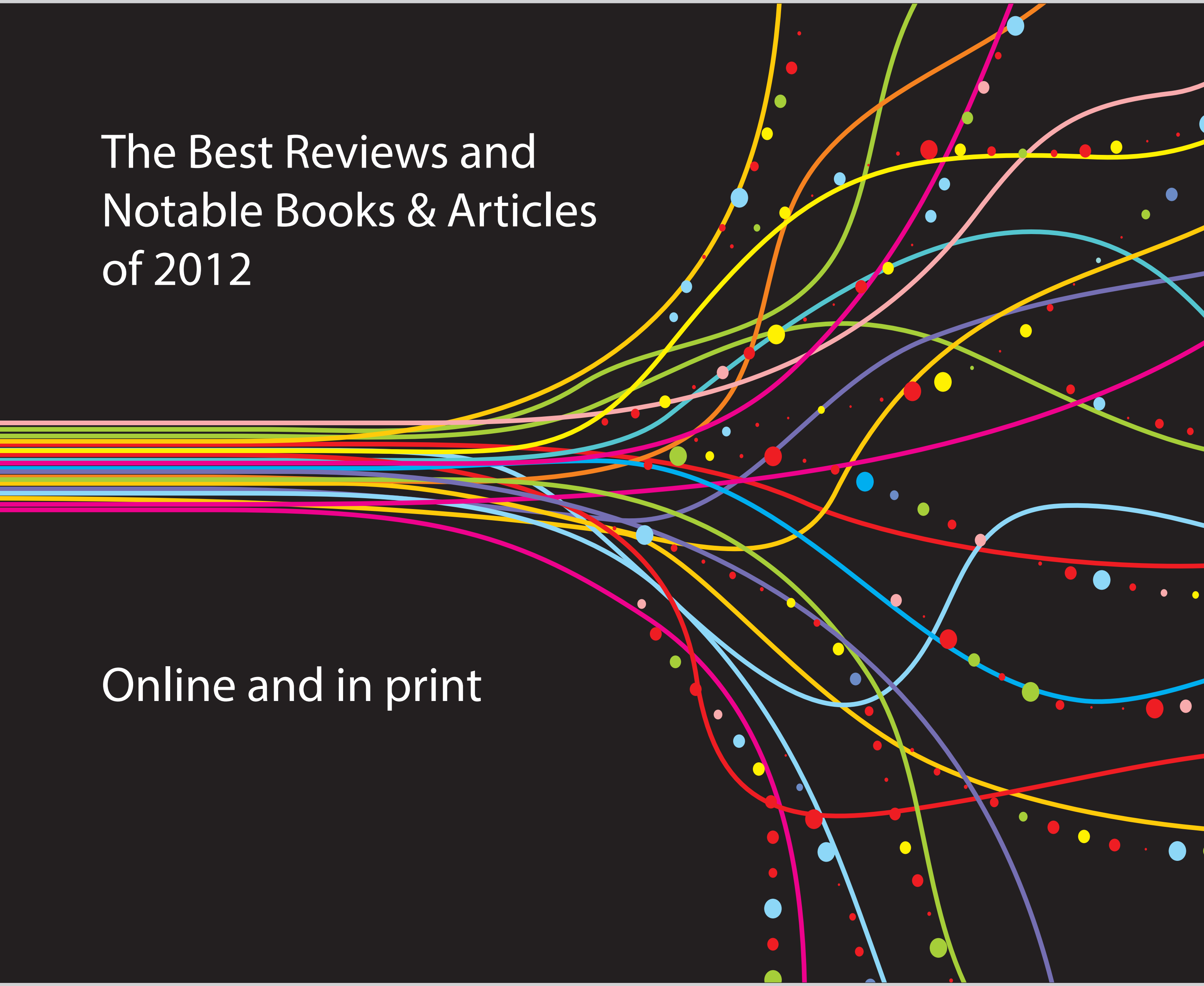
Rates: **$325.00 for six lines of text, 40 characters per line. $32.50 for each additional line after the first six. The MINIMUM is six lines.**

Deadlines: **20th of the month/2 months prior to issue date. For latest deadline info, please contact: *acmmediasales@acm.org***

Career Opportunities Online: **Classified and recruitment display ads receive a free duplicate listing on our website at: *http://jobs.acm.org***

Ads are listed for a period of 30 days.

For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmmediasales@acm.org



DOI:10.1145/2461256.2461280

Peter Winkler

Puzzled
Solutions and Sources

Last month (May 2013) we posed a trio of brainteasers concerning Ant Alice and her ant friends who always march at 1 cm/sec in whatever direction they are facing, reversing direction when they collide.

1. Time to fall.

This problem has been around for decades, with one version appearing in Francis Su's "Math Fun Facts" Web column at Harvey Mudd College (http://www.math.hmc.edu/funfacts). Recall that Alice is the middle ant of 25 ants on a meter-long stick, and the problem is to determine by what time she must have fallen off the end of the stick. The key observation—also useful in the other two puzzles—is that if you do not care about the identities of individual ants, you may as well think of them as passing through one another, instead of being bounced. That is, to an observer who thinks (mistakenly) all ants look alike, it appears that each ant simply walks from wherever he or she is to the end of the stick. Since this takes at most 100 seconds, it follows that every ant—including Alice—is off the stick by the time 100 seconds has passed.

2. Same order throughout.

In this problem the ants were placed randomly, and we want to determine the probability that Alice falls off the end she is facing initially. Now suppose, at the beginning, w ants face west, one being Alice. Then, at all subsequent times there will always be exactly w ants facing west (some of whom may have fallen off the west end), so w ants will ultimately fall off the west end of the stick. These ants will be exactly the first w ants counting from the west end at the beginning, since the ants stay in the same order throughout. It follows that Alice falls off the west end exactly when 13 or more ants were initially facing west, which occurs with probability approximately 58%. Since the same calculation applies if Alice initially faces east, the final answer is 58%.

3. There and back.

This problem takes place on a circle, one meter around, with only 12 ants, and the task is to determine the probability that Alice ends up where she started after 100 seconds. The first observation is that if we again ignore ant identities, it appears that each ant simply walks once around the circle. Thus, collectively, the ants' final 12 positions are the same as their initial 12 positions; the only question is whether Alice ends up at her own initial position or in some other ant's position. If the latter, the ants' positions must have rotated, since their cyclic order cannot change. But can this really happen?

Well, suppose k ants face clockwise initially, thus $12-k$ counterclockwise. We know this collective orientation will continue throughout. Viewing it physically, the ants will, by "conservation of angular momentum," always have a net clockwise momentum of $k-(12-k) = 2k-12$. If $k = 12$, then each ant will travel once clockwise around the circle, never colliding, and ending up where it started. If k=6, the net angular momentum will be zero, so again the ants must end up where they began; if k is strictly between 6 and 12, the ants must end up rotated out of position. Similar arguments apply when k is 6 or less, so we conclude that Alice comes back home exactly when k is 0, 6, or 12.

What is the probability that Alice ends up where she began? There are $2^{12} = 4{,}096$ ways to orient the ants, of which two have all the ants in one direction and "12 choose 6" have half the ants clockwise, so the final probability is (2 + (12 choose 6))/4,096, or approximately 22.6%.

All readers are encouraged to submit prospective puzzles for future columns to puzzled@cacm.acm.org.

Peter Winkler (puzzled@cacm.acm.org) is William Morrill Professor of Mathematics and Computer Science, at Dartmouth College, Hanover, NH.

[CONTINUED FROM P. 120] it's very relevant, because at the end of the day we constructed a theory of interaction, so whatever attracted us to this interactive thing was going to grow into a professional interest, as well.

You ended up having a common advisor, Manuel Blum.

SHAFI: The turning point was a course by Blum on computational number theory. At the end of the course, Blum asked this question about tossing a coin over the telephone. And somehow the idea that the combination of randomness, interaction and complexity of number theory problems could be used to emulate simultaneity in communication—to make it seem like flipping a coin on one end of the telephone and revealing it on the other hand happened at the same time rather than in succession—seemed unbelievably profound and exciting to me. And I think it's true about theoretical computer science in general… you use mathematics to solve real-world problems, but you're not really bound by the rules and conventions of classical mathematics.

The coin toss problem sounds similar to the game of mental poker that led to your 1983 paper on probabilistic encryption.

SHAFI: Mental poker had been posed before, but in that protocol, partial information could leak about the cards. We were working on the problem of how to play poker so that all partial information is hidden. I'm not really a card player; it was all very abstract. We had this idea of using quadratic residuosity, a hard problem from number theory, to code cards. So say the card is a seven of spades; we can think of its name as a binary string and represent each bit of this string as either a quadratic residue or Q-non-residue chosen at random. We proved that all partial information about the cards was hidden by this representation. It was almost an afterthought to say, "Wait a minute, there's a new public encryption scheme here where you can prove very strong security property; no partial information leaks."

"This often happens in mathematics—you start with something concrete and generalize, and in the end you get this beautiful theorem."

This often happens in mathematics—you start with something concrete and you generalize, and in the end you get this beautiful theorem. But you don't start by saying, "Let's think of a scheme that satisfies this security definition."

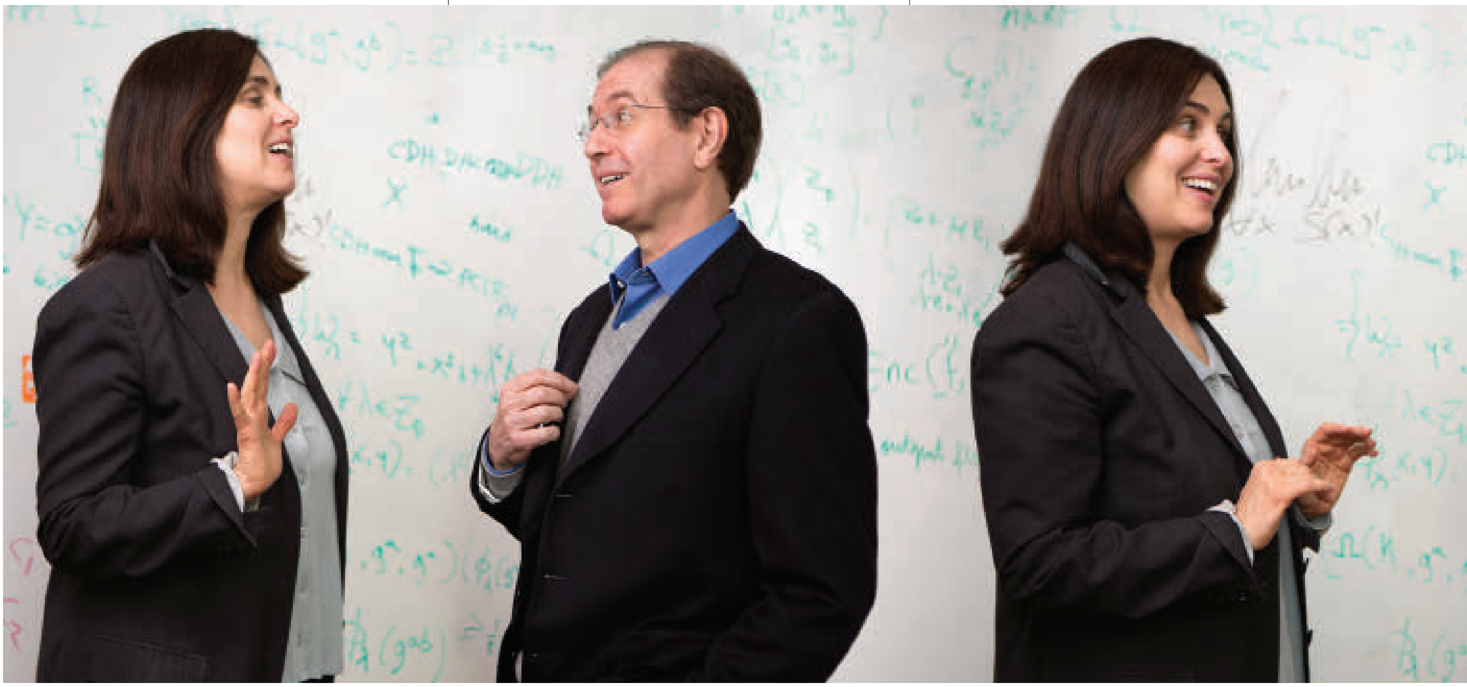
SILVIO: I agree that motivating examples are a big propeller for science. But we chose a very difficult problem that is a mixture of not only encryption, but also how you deal the cards after you encrypt them and how you make sure that the cards are getting a random shuffle. We were fearless, but we were also lucky.

SHAFI: In a sense, what people remember is probabilistic encryption. But there were also all these sub-contributions that made their way into later larger bodies of work on protocols and randomness.

One of the most powerful contributions was the notion of indistinguishability.

SILVIO: Computational indistinguishability roughly means that if you have limited computational power, being human, you cannot even distinguish between two things although they are very, very different from one another. In the context of encryption, this implies that if you are not the intended recipient of an encrypted message, not only can you not figure out the message

PHOTOGRAPHS BY BRYCE VICKMARK



in its entirety, but you don't have any inkling about its contents.

SHAFI: Nor can you figure out relationships between different messages.

Indistinguishability also played a role in your later work on zero knowledge interaction proofs and zero knowledge computations, where the notion of being unable to distinguish one reality from another is the key to analyzing secure protocols.

SILVIO: Well, first of all, leaving zero knowledge aside for a moment, what we created is a new kind of proof. Proofs are the most frustrating things. They're not fun to write and they're not fun to read. They slow you down. So we transformed them into a game. Say I claim a certain theorem to be true. Then I convince you that the following game has the following special property: if the theorem is true, I can win all the time. If the theorem is false, you win at least half of the time. Now we play, and I win. We play again, and I win again. Assume I win 20 times in a row. Then suddenly this very esoteric, long, tedious process of verifying becomes, if not fun, at least quick and interactive.

This is a transformation in two senses. First, since the proof is interactive, what convinces you is that you really played this game with me and you lost every single time. Here we are already gesturing toward zero knowledge, because that doesn't mean you can prove the theorem to somebody else. Second, there is this probability of error. We played 20 times, but maybe with a chance of one in a million, you would have not caught me if the theorem were false. But if we play 30 times, the chance is one in a billion. And if we play 300 times, the chance is one in the number of every elementary particle in the universe. So all of a sudden this probability is so miniscule that, for all practical purposes, it can be equated to zero.

"Proofs are the most frustrating things. They're not fun to write and they're not fun to read. They slow you down. So we transformed them into a game."

How did that lead to zero knowledge?

SILVIO: Zero knowledge interactive proofs are a way in which you can prove a theorem to me so that I know the theorem is correct, but I have no idea why. To accomplish this, you interact with me in a way such that if I knew the theorem were true, I could construct a virtual interaction with you that would be indistinguishable to me from the true interaction.

SHAFI: It's called the simulation paradigm. It was already in the probabilistic encryption paper as a proof method, but here it actually becomes part of the definition. If you think about this interview, the fact that you are talking to us convinces you of the fact that we are real. But beyond that, you could probably have surmised what we've said from all the papers we have written.

So a zero knowledge conversation is a conversation that could have been simulated so well that it would be indistinguishable from a real conversation.

SHAFI: That's right. If you can't distinguish between a true interactive proof and a simulated proof, you can conclude that the true interactive proof gave you nothing you couldn't have obtained yourself, besides knowing that your questions were actually answered by a real prover. So, the fact that in a true interactive proof the real prover answers your questions convinces you that a proof is correct but gives nothing else. C

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

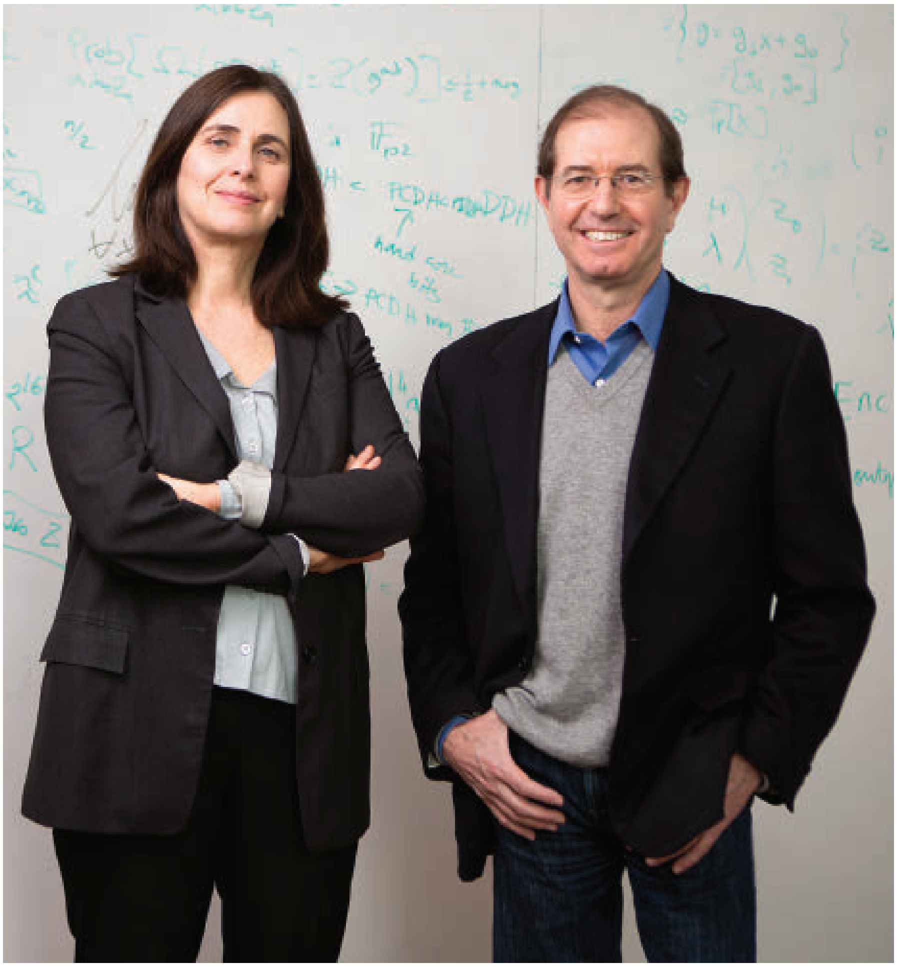
DOI:10.1145/2461256.2461281

Leah Hoffmann

Q&A
Cracking the Code

Turing Award recipients Shafi Goldwasser and Silvio Micali talk about proofs, probability, and poker.

THOUGH THEIR ROUTES to computer science differed, ACM A.M. Turing Award recipients Shafi Goldwasser and Silvio Micali have forged a common path in the field since they met in graduate school. Goldwasser was born in Israel and got hooked on programming in college at Carnegie Mellon University. Micali was born in Italy and discovered his interest in the field at the University of Rome through courses in lambda calculus and logic. Now both at MIT (Goldwasser holds a joint appointment at the Weizmann Institute of Science in Israel), the two have revolutionized cryptography by working through fundamental questions and forging a link with computational complexity. Since their groundbreaking 1983 paper on probabilistic encryption, their work has transformed the scope of cryptography from encrypting private messages to strengthening data security, facilitating financial transactions, and supporting cloud computing.



PHOTOGRAPH BY BRYCE VICKMARK

What drew you both to the field?

SILVIO: I started in physics and switched to mathematics. Then, toward the very end, I took two courses in discrete mathematics. So I switched to theoretical computer science and went to Berkeley, and that's where I met Shafi.

SHAFI: I went to college at Carnegie Mellon in applied mathematics. At the time, they didn't have an undergraduate degree in computer science, but there was a way to minor in computer science. When I graduated, I went to California for an internship at Rand as I was interested in AI, and one weekend I drove up with a friend to see Berkeley on a very sunny day. It was beautiful—green hills, bright blue skies—so off I went to Berkeley. At first, I was taking general courses, but then I ran into a group of theory students, one of whom was Silvio, and they sort of took me into their midst. The subject matter was appealing, but it was also a social thing—the theory students were an excited and an exciting bunch.

SILVIO: By contrast, when I landed at Berkeley, it was raining, and I discovered that I couldn't speak English. I knew there was a shuttle to campus, but I had to ask six people before they could grasp what I wanted. But things lightened up once we formed this band of brothers. This aspect Shafi mentioned about sociability,

[CONTINUED ON P. 118]

SENSE THE TRANSFORMATION

CALL FOR SUBMISSIONS

Digital Art | CG Research | Animation | Technological Innovations | Post Production | Interactive Applications | Mobile Graphics | Industry Trends

Present your creative achievements, innovations and stellar technical research.
Review program guidelines, plan your schedule and submit your best work.

Technical Papers	14 May
Courses	6 June
Symposium on Mobile Graphics and Interactive Applications	6 June
Art Gallery	13 June
Emerging Technologies	13 June
Computer Animation Festival	2 July
Posters	9 July
Technical Briefs	9 July

For complete details, visit **sa2013.siggraph.org/submitters**.

SIGGRAPH
ASIA 2013
HONG KONG

CONFERENCE 19 NOV - 22 NOV
EXHIBITION 20 NOV - 22 NOV

HONG KONG CONVENTION
AND EXHIBITION CENTRE

SA2013.SIGGRAPH.ORG

LEAD SPONSOR CREATEHK

SPONSORED BY acm