50TH ANNIVERSARY ISSUE

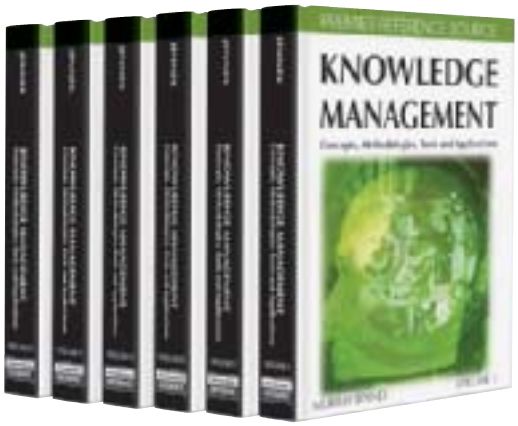
COMMUNICATIONS of the ACM

JANUARY 2008 VOLUME 51, NUMBER 1

Association for Computing Machinery

Stay Current with the Best Research in Technology

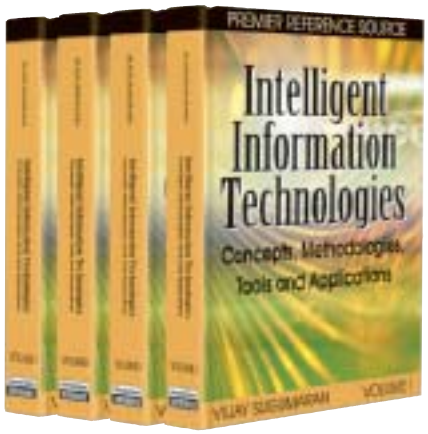
IGI Global's Multi-Volume Sets provide one-stop reference sources for the most influential and essential disciplines of information science and technology.



Knowledge Management
Concepts, Methodologies, Tools, and Applications

Murray Jennex, San Diego State University, USA

ISBN: 978-1-59904-933-5; 3,808 pp; © 2008 Available Now
US $1,950.00 hardcover; Online Access Only*: US $1,850.00



Intelligent Information Technologies
Concepts, Methodologies, Tools, and Applications

Vijayan Sugumaran, Oakland University, USA

ISBN: 978-1-59904-941-0; 2,610 pp; © 2008 Available Now
US $1,550.00 hardcover; Online Access Only*: US $1,450.00

Additional Titles Available from

IGI GLOBAL
DISSEMINATOR OF KNOWLEDGE®

Formerly Idea Group Inc.

FREE Institutional Online Access!*

Artificial Intelligence for Advanced Problem Solving Techniques
Dimitris Vrakas and Ioannis Pl. Vlahavas, Aristotle University, Greece
ISBN: 978-1-59904-705-8
300 pp; © 2008 Available Now
US $180.00 hardcover
Online Access Only*: US $132.00

Dictionary of Information Science and Technology
Mehdi Khosrow-Pour, Information Resources Management Association, USA
ISBN: 978-1-59904-385-2
1,016 pp; © 2007 Available Now
US $495.00 hardcover
Online Access Only*: US $450.00

Data Mining Patterns: New Methods and Applications
Pascal Poncelet, Ecole des Mines d'Ales, France, Florent Masseglia, Projet AxIS-INRIA, France, and Maguelonne Teisseire, Universite Montpellier, France
ISBN: 978-1-59904-162-9
323 pp; © 2008 Available Now
US $180.00 hardcover
Online Access Only*: US $132.00

Agile Software Development Quality Assurance
Ionnis G. Stamelos, Aristotle University of Thessaloniki, Greece, and Panagiotis Sfetsos, Alexander Technological Educational Institution of Thessaloniki, Greece
ISBN: 978-1-59904-216-9
268 pp; © 2007 Available Now
US $165.00 hardcover
Online Access Only*: US $132.00

Handbook of Computational Intelligence in Manufacturing and Production Management
Dipak Laha, Jadavpur University, India, and Purnendu Mandal, Lamar University, USA
ISBN: 978-1-59904-582-5
516 pp; © 2008 Available Now
US $210.00 hardcover
Online Access Only*: US $156.00

Encyclopedia of Portal Technologies and Applications
Arthur Tatnall, Victoria University, Australia
ISBN: 978-1-59140-989-2
1,308 pp; © 2007 Available Now
US $565.00 hardcover
Online Access Only*: US $452.00

Recommend a Title to Your Librarian Today!

*Free online access to Information Science Reference titles is for institutions only and excludes the Dictionary of Information Science and Technology.

View more titles from IGI Global at www.igi-global.com.
IGI Global • 701 E. Chocolate Ave., Suite 200 • Hershey, PA 17033-1240 USA • 1-866-342-6657
cust@igi-global.com • www.igi-global.com

CALL FOR PAPERS

2008 ACM International Conference on Computing Frontiers

May 5 - 7, 2008, Ischia, Italy

Sponsored by ACM - SIGMICRO

GENERAL CHAIR
Alex Ramirez, UPC, ES

PROGRAM CO-CHAIRS
Gianfranco Bilardi, Università di Padova, IT
Michael Gschwind, IBM TJ Watson, US

STEERING COMMITTEE
Monica Alderighi, INAF, IT
Utpal Banerjee, Intel, US
Steven Beaty, Metro State College Denver, US
Sergio D'Angelo, INAF, IT
Michel Dubois, U. of Southern California, US
Kemal Ebcioglu, Global Supercomputing, US
Gearold Johnson, Colorado State U., US
Sally A. McKee, Cornell University, US
Cecilia Metra, University of Bologna, IT
José Moreira, IBM TJ Watson, US
Valentina Salapura, IBM TJ Watson, US
Giacomo Sechi, INAF, IT
Per Stenström, Chalmers U. of Technology, SE

FINANCE/REGISTRATION CHAIR
Carsten Trinitis, TU München, DE

SPECIAL SESSIONS CHAIR
Osman Unsal, BSC, ES

LOCAL ARRANGEMENTS CHAIR
Claudia Di Napoli, CNR, IT

PUBLICITY CHAIR
Julita Corbalan, UPC, ES

LIAISON CHAIR FOR ASIA
Hitoshi Oi, University of Aizu, JP

PUBLICATION CHAIR
Sergio D'Angelo, INAF, IT

WEB CHAIR
Greg Bronevetsky, LLNL, US

The increasing needs of present and future computation-intensive applications have stimulated research in new and innovative approaches to the design and implementation of high-performance computing systems. These boundaries between state of the art and revolutionary innovation constitute the computing frontiers that must be pushed forward to provide the computational support required for the advancement of all science domains and applications. This conference focuses on a wide spectrum of advanced technologies and radically new solutions; it is designed to foster communication among many scientific and technological disciplines.

Authors are invited to submit papers on all areas of innovative computing systems that extend the current frontiers of computer science and engineering and that will provide advanced systems for current and future applications.

Papers are sought on theory, methodologies, technologies, and implementations concerned with innovations in computing paradigms, computational models, architectural paradigms, computer architectures, development environments, compilers, and operating environments. Papers are solicited in, but not limited to, the following areas:

- Non-conventional computing
- Next-generation high performance computing, esp. novel high-performance systems (including Cell, GPGPU and custom accelerators)
- Applications, programming models and performance analysis of parallel architectures and novel high-performance systems
- Virtualization and virtual machines
- Grid computing
- Compilers and operating systems
- Workload characterization of emerging applications
- Service oriented architecture (SOA) and system impact
- Supercomputing
- SOC architectures, embedded systems and special-purpose architectures
- Temperature, energy, and variability-aware designs; fault tolerance and reliability
- System management and security
- Quantum and nanoscale computing
- Computational biology
- Reconfigurable computing and architecture prototyping
- Autonomic and organic computing
- Computation intelligence frontiers: theory and industrial applications

All papers will be published in the conference proceedings and will be made available in the ACM Digital Library. In addition, selected papers will be invited to appear in a special issue of the HiPEAC journal.

Submitted manuscripts should not exceed 20 double-spaced, single-column pages, including figures, tables, and references. Submission implies that at least one author will register for the conference and present the paper, if accepted. Submissions must be made electronically as Adobe PDF files through the conference web-site, and must not be simultaneously submitted to any other publication outlet.

Important Dates

Paper submission deadline:	Author notification:	Final papers due:
December 7, 2007	January 18, 2008	February 22, 2008

Table of Contents

January 2008

50

From the Editor's Desk

Voices

Departments

Articles

Breakthrough Research: A Preview of Things to Come

Columns

13

17

DIS2008

CAPE TOWN · SOUTH AFRICA

designing interactive systems

25-27 February 2008

www.sigchi.org/dis2008

SIGCHI

UNIVERSITY OF CAPE TOWN

acm Association for Computing Machinery

NOKIA Connecting People

SAP

Microsoft

DESIGN INDABA

Editorial Pointers

The first issue was 20 pages. Its technical content fell into three departments: Standards; Computer Techniques; and Unusual Applications. It told of new methods for square-root computations and of a programmed binary counter for the IBM Type 650 calculator.

It was January 1958. Sputnik had been launched by the Soviet Union three months prior; the computing field was still abuzz over IBM's commercial release of its pioneering Fortran compiler; and two teens in Liverpool named McCartney and Lennon had recently been introduced by mutual friends.

Communications of the ACM would join the then four-year-old *Journal of the ACM* as partners in the dissemination of computing research news. Indeed, CACM was created as a vehicle for ACM members to "communicate" their research findings with each other in an effort to eliminate reinventing the wheel. Over the years, this membership would grow to represent every known computing field; from every corner of the world, from the halls of academia to corporate suites and entrepreneurial garages. It would become ACM's flagship publication and share the kind of landmark discoveries, research, and accomplishments that would influence the course of computer science and, in turn, our daily lives and livelihoods.

Now it's 50.

To honor this milestone, we've invited many leading contributors from CACM's past, present, and future to share their memories and perceptions. The Editors-in-Chief who have guided this publication through the decades played a great role in CACM's distinguished history as chronicler of the field. We are honored to present their recollections here. We are also proud to introduce the newly appointed EIC, Moshe Vardi, who details the next editorial direction for CACM and the planning and preparation that has led up to this model (see page 44).

We also called on prominent voices in the computing field (and in the pages of CACM over the years) to participate in this anniversary celebration by sharing their thoughts, insights, and concerns about some of the next big stories likely to emerge. We are indebted to all who have offered their efforts and expertise to the issue. Their participation truly reflects great respect for this publication and for the audience it continues to serve.

On a personal note, many thanks to CACM's Senior Editor Andrew Rosenbloom; Managing Editor Tom Lambert; and Art Director Caren Rosenblatt for an extraordinary team effort.

We hope these stories serve to inspire, influence, and teach—a legacy befitting ACM's premier publication. Enjoy.

Diane Crawford

Editor

COMMUNICATIONS of the ACM

A monthly publication of
the ACM Publications Office

ACM
2 Penn Plaza, Suite 701
New York, NY 10121-0701 USA
(212) 869-7440 FAX: (212) 869-0481

Group Publisher: Scott Delman

Editor: Diane Crawford
Managing Editor: Thomas E. Lambert
Senior Editor: Andrew Rosenbloom
Editorial Assistant: Zarina Strakhan
Copyright: Deborah Cotton

Contributing Editors
Phillip G. Armour; Hal Berghel; Michael A. Cusumano; Peter J. Denning; Robert L. Glass; Seymour Goodman; Rebecca Mercuri; Peter G. Neumann; Pamela Samuelson; Meg McGinity Shannon

Art Director: Caren Rosenblatt
Production Manager: Lynn D'Addesio

Advertising
ACM Advertising Department
2 Penn Plaza, Suite 701, New York, NY 10121-0701
(212) 869-7440; Fax: (212) 869-0481
Account Executive:
William R. Kooney: acm-advertising@acm.org

For the latest media kit—including rates—contact:
Graciela Jacome: jacome@acm.org

Contact Points
CACM editorial: crawford_d@acm.org
Copyright permission: permissions@acm.org
Calendar items: calendar@acm.org
Change of address: acmcoa@acm.org

Communications of the ACM
(ISSN 0001-0782) is published monthly by the ACM, 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

Printed in the U.S.A.

Association for Computing Machinery

Advancing Computing as a Science & Profession

COMMUNICATIONS OF THE ACM • *A monthly publication of the ACM Publications Office*

ACM, 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA (212) 869-7440 FAX: (212) 869-0481

(A Look Back at) Go To Statement Considered Harmful

Edsger Dijkstra wrote a Letter to the Editor of *Communications* in 1968, criticizing the excessive use of the **go to** statement in programming languages. Instead, he encouraged his fellow computer scientists to consider structured programming. The letter, originally entitled "A Case Against the Goto Statement," was published in the March 1968 issue under the headline "Go To Statement Considered Harmful." It would become the most legendary CACM "Letter" of all time; "Considered Harmful" would develop into an iconic catch-all. Dijkstra's comments sparked an editorial debate that spanned these pages for over 20 years. In honor of the occasion, we republish here the original letter that started it all.

PAUL WATSON

Editor:

For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of **go to** statements in the programs they produce. More recently I discovered why the use of the **go to** statement has such disastrous effects, and I became convinced that the **go to** statement should be abolished from all "higher level" programming languages (i.e. everything except, perhaps, plain machine code). At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so.

My first remark is that, although the programmer's activity ends when he has constructed a correct program, the process taking place under control of his program is the true subject matter of his activity, for it is this process that has to accomplish the desired effect; it is this process that in its dynamic behavior has to satisfy the desired specifications. Yet, once the program has been made, the "making" of the corresponding process is delegated to the machine.

My second remark is that our intellectual powers are rather geared to master static relations and that our powers to visualize processes evolving in time are relatively poorly developed. For that reason we should do (as wise programmers aware of our limitations) our utmost to shorten the conceptual gap between the static program and the dynamic process, to make the correspondence between the program (spread out in text space) and the process (spread out in time) as trivial as possible.

Let us now consider how we can characterize the progress of a process. (You may think about this question in a very concrete manner: suppose that a process, considered as a time succession of actions, is stopped after an arbitrary action, what data do we have to fix in order that we can redo the process until the very same point?) If the program text is a pure concatenation of, say, assignment statements (for the purpose of this discussion regarded as the descriptions of single actions) it is sufficient to point in the program text to a point between two successive action descriptions. (In the absence of **go to** statements I can permit myself the syntactic ambiguity in the last three words of the previous sentence: if we parse them as "successive (action descriptions) "we mean successive in text space; if we parse as "(successive action) descriptions" we mean successive in time.) Let us

call such a pointer to a suitable place in the text a "textual index."

When we include conditional clauses (**if** *B* **then** *A*), alternative clauses (**if** *B* **then** *A*1 **else** *A*2), choice clauses as introduced by C.A.R. Hoare (case[i] of (*A*1, *A*2, ... , *An*)), or conditional expressions as introduced by J. McCarthy (*B*1→*E*1, *B*2 → *E*2, ... , *Bn* → *En*), the fact remains that the progress of the process remains characterized by a single textual index.

As soon as we include in our language procedures we must admit that a single textual index is no longer sufficient. In the case that a textual index points to the interior of a procedure body the dynamic progress is only characterized when we also give to which call of the procedure we refer. With the inclusion of procedures we can characterize the progress of the process via a sequence of textual indices, the length of this sequence being equal to the dynamic depth of procedure calling.

Let us now consider repetition clauses (like, **while** *B* **repeat** *A* or **repeat** *A* **until** *B*). Logically speaking, such clauses are now superfluous, because we can express repetition with the aid of recursive procedures. For reasons of realism I don't wish to exclude them: on the one hand, repetition clauses can be implemented quite comfortably with present day finite equipment; on the other hand, the reasoning pattern known as "induction" makes us well equipped to retain our intellectual grasp on the processes generated by repetition clauses. With the inclusion of the repetition clauses textual indices are no longer sufficient to describe the dynamic progress of the process. With each entry into a repetition clause, however, we can associate a so-called "dynamic index," inexorably counting the ordinal number of the corresponding current repetition. As repetition clauses (just as procedure calls) may be applied nestedly, we find that now the progress of the process can always be uniquely characterized by a (mixed) sequence of textual and/or dynamic indices.

The main point is that the values of these indices are outside programmer's control; they are generated (either by the write-up of his program or by the dynamic evolution of the process) whether he wishes or not. They provide independent coordinates in which to describe the progress of the process.

Why do we need such independent coordinates? The reason is—and this seems to be inherent to sequential processes—that we can interpret the value of a variable only with respect to the progress of the process. If we wish to count the number, *n* say, of people in an initially empty room, we can achieve this by increasing *n* by one whenever we see someone entering the room. In the in-between moment that we have observed someone entering the room but have not yet performed the subsequent increase of *n*, its value equals the number of people in the room minus one!

The unbridled use of the **go to** statement has an immediate consequence that it becomes terribly hard to find a meaningful set of coordinates in which to describe the process progress. Usually, people take into account as well the values of some well chosen variables, but this is out of the question because it is relative to the progress that the meaning of these values is to be understood! With the **go to** statement one can, of course, still describe the progress uniquely by a counter counting the number of actions performed since program start (viz. a kind of normalized clock). The difficulty is that such a coordinate, although unique, is utterly unhelpful. In such a coordinate system it becomes an extremely complicated affair to define all those points of progress where, say, *n* equals the number of persons in the room minus one!

The **go to** statement as it stands is just too primitive; it is too much an invitation to make a mess of one's program. One can regard and appreciate the clauses considered as bridling its use. I do not claim that the clauses mentioned are exhaustive in the sense that they will satisfy all needs, but whatever clauses are suggested (e.g. abortion clauses) they should satisfy the requirement that a programmer independent coordinate system can be maintained to describe the process in a helpful and manageable way.

It is hard to end this with a fair acknowledgment. Am I to judge by whom my thinking has been influenced? It is fairly obvious that I am not uninfluenced by Peter Landin and Christopher Strachey. Finally I should like to record (as I remember it quite distinctly) how Heinz Zemanek at the pre-ALGOL meeting in early 1959 in Copenhagen quite explic-

itly expressed his doubts whether the **go to** statement should be treated on equal syntactic footing with the assignment statement. To a modest extent I blame myself for not having then drawn the consequences of his remark.

The remark about the undesirability of the **go to** statement is far from new. I remember having read the explicit recommendation to restrict the use of the **go to** statement to alarm exits, but I have not been able to trace it; presumably, it has been made by C.A.R. Hoare. In [1, Sec. 3.2.1.] Wirth and Hoare together make a remark in the same direction in motivating the case construction: "Like the conditional, it mirrors the dynamic structure of a program more clearly than **go to** statements and switches, and it eliminates the need for introducing a large number of labels in the program."

In [2] Guiseppe Jacopini seems to have proved the (logical) superfluousness of the **go to** statement. The exercise to translate an arbitrary flow diagram more or less mechanically into a jumpless one, however, is not to be recommended. Then the resulting flow diagram cannot be expected to be more transparent than the original one.

References

1. Wirth, Niklaus, and Hoare, C.A.R. A contribution to the development of ALGOL. *Comm. ACM 9* (June 1966), 413–432.
2. Bohn, Corrado, and Jacopini, Guiseppe. Flow Diagrams, Turing machines and languages with only two formation rules. *Comm. ACM 9* (May 1966) 366–371.

Edsger W. Dijkstra
Technological University
Eindhoven, The Netherlands

Communications of the ACM
March 1968, Vol. 11, No. 3, pg 147

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

Dear Colleague,

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

As an ACM member you have access to leading-edge resources including publications and conferences covering the entire computing field. Whether you are pursuing scholarly research, building systems and applications, or managing computing projects, ACM offers opportunities to advance your interests.

MEMBER BENEFITS INCLUDE:

- Access to ACM's **new Career & Job Center** offering a host of exclusive career-enhancing benefits
- **Free e-mentoring services** provided by MentorNet®
- **Free and unlimited access to 2,200 online courses** from SkillSoft®
- **Free and unlimited access to 600 online books** from Safari® Books Online, featuring leading publishers, including O'Reilly (Professional Members only)
- **Free and unlimited access to 500 online books** from Books24x7®
- A subscription to ACM's flagship monthly magazine, ***Communications of the ACM***
- The option to subscribe to the full **ACM Digital Library**
- The **Guide to Computing Literature**, with over one million searchable bibliographic citations
- The option to connect with the **best thinkers in computing** by joining **34 Special Interest Groups** or **hundreds of local chapters**
- **ACM's 40+ journals and magazines** at special member-only rates
- ***TechNews***, ACM's tri-weekly email digest delivering stories on the latest IT news
- ***CareerNews***, ACM's bi-monthly email digest providing career-related topics
- ***MemberNet***, ACM's e-newsletter, covering ACM people and activities
- **Email forwarding service & filtering service**, providing members with a free acm.org email address and high-quality **Postini** spam filtering
- And much, much more!

ACM's worldwide network of more than 83,000 members range from students to seasoned professionals and includes many of the leaders in the field. ACM members get access to this network and the advantages that come from their expertise to keep you at the forefront of the technology world.

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

Sincerely,

Stuart I. Feldman

Stuart I. Feldman
President
Association for Computing Machinery

Association for Computing Machinery

Advancing Computing as a Science & Profession

Association for Computing Machinery

Advancing Computing as a Science & Profession

membership application & *digital library* order form

Priority Code: ACACM28

You can join ACM in several easy ways:

Online
http://www.acm.org/join

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

Fax
+1-212-944-1318

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

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

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

Please print clearly

Name

Address

City | State/Province | Postal code/Zip

Country | E-mail address

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

Purposes of ACM

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

I agree with the Purposes of ACM:

Signature

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

choose one membership option:

PROFESSIONAL MEMBERSHIP:

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

STUDENT MEMBERSHIP:

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

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

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

RETURN COMPLETED APPLICATION TO:

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

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

Satisfaction Guaranteed!

payment:

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

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

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

❑ ACM Digital Library ($99) $ _____

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

Total Amount Due $ _____

Card # | Expiration date

Signature

The Best Place to Find the Perfect Job... Is Just a Click Away!

No need to get lost on commercial job boards.
The ACM Career & Job Center is tailored specifically for you.

JOBSEEKERS

- Manage your job search
- Access hundreds of corporate job postings
- Post an anonymous resume
- Advanced Job Alert system

EMPLOYERS

- Quickly post job openings
- Manage your online recruiting efforts
- Advanced resume searching capabilities
- Reach targeted & qualified candidates

NEVER LET A JOB OPPORTUNITY PASS YOU BY!
START YOUR JOB SEARCH TODAY!

http://www.acm.org/careercenter

Association for Computing Machinery
Advancing Computing as a Science & Profession

POWERED BY JOBTARGET

Phillip G. Armour

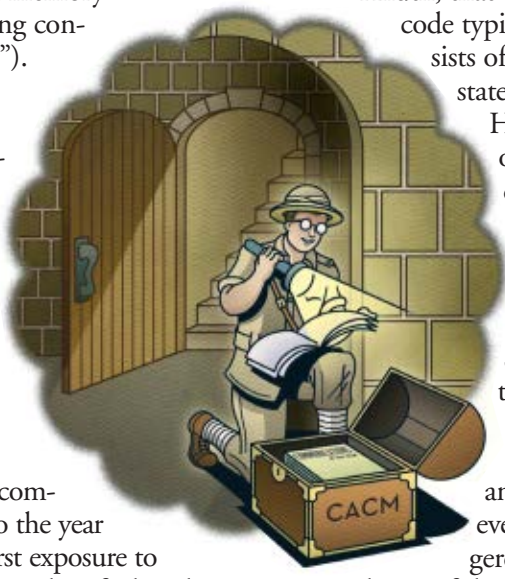
Digging CACM

An archeological view of a classic.

Archeology is an interesting, if somewhat dusty, science. Archeologists dig into the ground to unearth artifacts from antiquity and from their age-encrusted appearance and structure try to deduce their purpose and better understand the civilizations that created them.

And so it was one evening when I led an expedition down into the stygian depths of my basement in search of the earliest edition of *Communications of the ACM*. Amid rows of yellowing *National Geographic* magazines, boxes of ancient canceled checks, and fading but still blue copies of SIGSOFT proceedings I unearthed the oldest known (to me) copy of *Communications*—it was the February 1985 issue.

By the time I encountered CACM it was already a highly evolved organism. The earliest known direct ancestor of today's magazine has been carbon dated as far back as 1958. This proto-CACM included such advances in computation as a (language-free) implementation of Newton's square-root formula and the use of tables to calculate a least-squares approximation. It also included a binary counter for use with an IBM 650 calculater (sic) including the actual core locations, the decimal machine instructions (in Linear-C?), and the explicit branch memory addresses (this was before "direct program memory branch addressing considered harmful"). The article also included a delightful hand-written flow-chart of the process—the software engineering equivalent of the Lascaux cave paintings.



Fast-forward through several computing epochs to the year 1985 and my first exposure to CACM, at least as identified in the historical record. The articles in this February 1985 edition give us examples of how the business of software has changed and how it has remained the same.

Programming Pearls. In the "Programming Pearls" column, we learn of a programmer who can log in while seated but gets locked out if he stands up. There is an international banking system that stops dead when someone enters the capital of Ecuador. And we also find, courtesy of the research of the indomitable Donald Knuth, that 70% of code typically consists of only four statement types. Has any of this changed? My unscientific observation of modern systems is that weird defects have not become an extinct or even endangered species, and even if the actual number of statement types used remains quite small, the job is still to get them right.

Grosch's Law inferred that the cost of a computer will rise as the square root of its power. The fastest IBM mainframe at that time, a 3081, could cycle around 10 MIPS and cost a mere $3.7

PETERHOEY

million. Given that today's Xbox can hit 6,400 MIPS it should therefore cost around $93 million. Clearly, Moore's Law has trumped Grosch's Law.

Readability. In a study on the readability of major computing periodicals, CACM scored in the "Difficult" range of the Flesch Reading Ease Index. It scored higher (more readable) than any of its major competitors at the time—the most difficult to read being the *IBM Systems Journal*, which was rated as solidly in the "Very Difficult" category. Of course, some elements in our profession believe that if something is relatively easy to read it cannot be worthwhile. However, when the job of a magazine is to communicate it would seem that any effort put into making that communication more efficient would be valuable. I think CACM continues to invest heavily in this area. Certainly these days it is much easier to read than some of the 1985 benchmarks such as-then-number-one in circulation *Infosystems*, number two in circulation *Datamation*,[1] number three in circulation *Computer Decisions* and the much-missed number four *Mini-Micro Systems*. Clearly there has been a Darwinian process at work.

Niklaus Wirth won the ACM A.M. Turing Award in 1984. In his address published in the February 1985 issue, he discusses his pursuit of an "appropriate formalism" in programming languages and describes some of the common characteristics of the projects that informed his view of languages at the time. They included the idea of a bootstrap tool that provided capability but more importantly acted as the basis for the next generation or iteration. He identified the need to separate requirements and capabilities into the essential and the "nice to have." He thought the choice of tools important but that they must not require more effort to learn than they save. And, most importantly, he viewed each project primarily as a learning experience supported by the most essential (though "elusive and subtle") element of an enthusiastic team that collectively believes in the worth of the endeavor. Well *that* hasn't changed.

Selecting Decision Support System (DSS) Software was a cogent and useful article on how to select DSS including, if I perceived correctly, screen shots from Lotus 1-2-3. The striking thing is that none of these systems exist now, though some have probably evolved into today's offerings. The evaluation criteria with a few deletions (such as the ability to produce "basic plots and charts") could serve as a starting point for an assessment capability today. Curiously, none of them mention service-oriented architecture or even Web access.

The article really addresses a process for selection rather than what is being selected. While the target software has changed considerably, much of the process for selecting it could still apply. This just goes to show that software and systems come and go, but process lives forever.

Programmer Productivity. Starts off with a complaint about the dearth of empirical research in the kinds of tools that really help programmers. The list of software tools desired by programmers in 1985 shows how far we've come in this area (screen editor anyone?)

Event Simulation Modeling Language. With 56 cited articles, this article on a condition specification (CS) language as part of a model specification (MS) language could have been written yesterday. It doesn't seem that this aspect of software engineering and computer science has advanced as it should given such a start.

Efficiency of List Update and Paging Rules. This article alone must have pulled the magazine's Flesch Reading Ease Index down by a good 20 points. Well, it was in the "Research Contributions" section, so that would explain the six theorems and their associated proofs embedded in the article. While not everyone's cup of $\sum t_i$ it showed the breadth of topics in this magazine and might well have appealed to those also interested in the…

...Positions Vacant in Computer Science. I counted 186 positions in computer science and software engineering in universities and colleges around the world (but particularly around the U.S.), 31 calls for papers just for February 1985–May 1985,

[1] *Datamation* still exists, though not in paper form; see www.datamation.com.

and 184 conferences, symposia, workshops, expositions, and general meetings through early 1986. Clearly, a lot was going on in 1985 and CACM was keeping readers up to date on what that was. Perhaps today the academic appointment market is not quite as brisk as it was, but there are still many conferences, expositions and, yes, meetings.

Like analyzing dinosaur bones, sampling one edition of a magazine with a history as long as *Communications of the ACM* in a discipline with a history as short as software engineering only provides a snapshot of what is an evolutionary process. The business of software and CACM (and to some extent myself personally) have grown up together. Whenever we grow and evolve, we can always look back a few years and see ideas and approaches that seem dated and strange now. But we can also see things that truly communicated what was going on at that time in our profession, located our knowledge in a pertinent, topical, and readable form and also pointed to the future of our profession.

It is interesting and instructive to look back and see where we've come from. But it's even more interesting to ponder what is coming, and it is good to know that, whatever it is, *Communications of the ACM* will be there communicating it to us. C

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

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

New Horizons

ACM SIGGRAPH launches the premiere SIGGRAPH Asia in Singapore

Programs include:

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

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

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

Queries?

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

SIGGRAPHASIA2008

Conference and Exhibition on Computer Graphics and Interactive Techniques

Singapore, 10-13 December 2008

Held in

UNIQUELY Singapore
visitsingapore.com

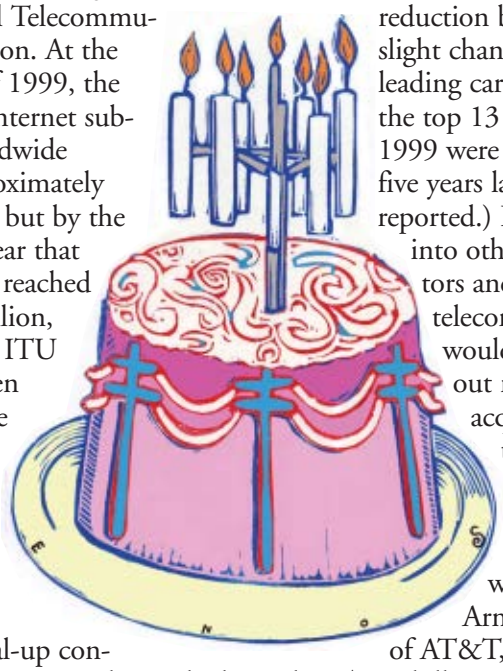
Happy Anniversary, CACM

Sampling the various slices of the telecommunications and computing spectrum over the years.

When I started writing the "Staying Connected" column in 1999, the U.S. Telecommunications Act had existed for just three years. The column's mission was to inform about the changes in the telecom arena and to serve as an acknowledgment that telecommunications and computers had irrevocably collided.

At that time, telecom companies were fighting over the customer and the bill, and to get their particular brand out front, they wanted to offer local service. The walls had crumbled between the distinctions of what services each provider could and would offer. The traditional long-distance companies, for instance, were trying to offer up not just long-distance voice, but all communication services, like local, wireless, Internet and even cable TV, in one bundle with one all-encompassing bill. Cable companies, which had a direct path to the consumer, attracted attention for the traditional voice companies. U.S. wireless subscribership was ascending, a strong 69 million, but was still merely a drop in the bucket compared to today's 233 million.

It was also during this year, in 1999, that the Internet came of age, at least according to the International Telecommunication Union. At the beginning of 1999, the number of Internet subscribers worldwide totaled approximately 150 million, but by the end of the year that number had reached over 250 million, according to ITU data. But even with all those logging on, most in the U.S.—97% say some reports—were gaining access via dial-up connections. America Online, which was yet to merge with Time Warner, had brought the Internet to the general public to the tune of 18 million subscribers.



The past decade has proven the telecom arena industrious and resilient. The Gartner Group reported that the telecom market's capitalization dropped by about $1 trillion following the dot-com crash in early 2000, but this reduction brought only slight change to list of leading carriers. (Ten of the top 13 carriers in 1999 were still up there five years later, Gartner reported.) In order to get into other business sectors and stay alive, telecom companies would either build out networks or acquire competitors, feeding into the merger frenzy. (Remember when Michael Armstrong, head of AT&T, spent more than $100 billion in acquisitions—mainly for cable properties—to "bring this communications revolution to American families"?)

During these tumultuous years

RANDALL ENOS

for telecom it wasn't just the services they were providing that changed dramatically. Traditional carriers had to reinvent the business model toward one of enabling communication to where a person is rather than to a place he might be. Gargantuan companies had to compete with sleeker, more innovative players. Telecom players were forced to acquire smaller companies that could broaden their reach, broaden their menu of services. Soon they were trumpeting the triple and quadruple play—voice, video, data, and mobility.

Telecom's metamorphosis in that short time is, of course, part of a bigger more steady change the industry has undergone over the past 50 years. It is especially relevant as we look back at CACM and celebrate the publication's own anniversary. Interwoven with the progress and evolution of telecommunications is the progress and evolution of the computer. This intermingling between the two disciplines will only grow stronger. The role computers have played not only in telecom's development but in society's maturity brings us back to ACM and this publication.

The Columbia Connection

The origins of ACM sound almost quaint in retrospect, but the momentum one meeting created would help foster world change. Some 60 years ago, a bunch of folks interested in computing ideas expressed a desire to get together and exchange discoveries with their peers. It might be application techniques or standards that would be discussed or debated, but the way accomplished mathematician and CACM author Franz Alt described it in a column, it was more like just 78 people coming together at New York City's Columbia University trying to grapple with new findings and hoping they'd meet other people as interested in the computing field, and in furthering the computer field, as they were.

Wonder if there was an awareness of the kind of frontier that lay ahead of those people as they filed into that prestigious building way back in 1947. They came up with a name for their organization, Eastern Association for Computing Machinery, and set about advancing the cause of computing. Of course, Eastern would later be dropped from the title and the group of interested folks from both the academic and professional worlds of computing would grow from 78 to today's more than 80,000 members.

Only a year before the organization was formed, in 1946, the ENIAC or Electronic Numerical Integrator and Computer, which derived from a military need to help soldiers determine settings on weapons, was developed. The deciphering was based on complex calculations that proved too time consuming for humans. While strategizing for the war, the calculator was being tinkered with at the University of Pennsylvania. The finished product stood bloated with 19,000 vacuum tubes, tipped the scales at more than 30 tons, and used almost 200 kilowatts of electrical power each day, according to reports.

"By today's standards for electronic computers the ENIAC was a grotesque monster," wrote Martin Weik of the Ballistic Research Laboratories at Aberdeen Proving Ground back in 1961. Still, the ENIAC became the prototype for most other modern computers. A 1958 report titled *Defense Spending and the U.S. Economy* that came from the operations research office of the John Hopkins University tagged it "the first modern electronic computer."

In the early 1950s, again from a military need, IBM developed and distributed its own "defense calculator." In 1953, 19 of these IBM machines, later renamed the "701" were sold, according to Chronology of Digital Computing Machines. Two of these units went to the defense department.

Within that next decade, academic science projects, also originating in part from defense contracts, would attract the attention of researchers. Plans for ARPA, or Advanced Research Projects Agency, had been laid. Some of the primary researchers involved learned of other scientists' compatible work at an ACM conference held in 1967, according to reports. These projects would evolve into what we know now as the Internet.

The ACM had understood the need to record and review literature that bubbled from these

Through the years, CACM held a forum for computer folks who were looking for advice on how to bridge the divide between researchers in the labs and those who were implementing programming, and the other bridge between the developers and the end users.

computing pioneers in those early days, and a journal was born. This evolved into *Communications of the ACM* debuting in 1958. That 1958 is a lifetime ago in computer-speak only reminds us of the more innocent era this publication was born into. The average home in the U.S., after all, cost a paltry $12,220 and a gallon of gas was 24 cents.

Research Plus

Communications of the ACM has chronicled the astonishing developments in the computer industry. In the pages of CACM, the progress and change and possibility are dissected and explored. Click through the archives of this publication, and you'll find celebrities for the tech-obsessed, like Leonard Kleinrock co-writing "A Study of Line Overhead in the Arpanet" in the 1970s, or a tribute for the late Jonathan Postel, who helped create and run the Internet and direct the Internet Assigned Numbers Authority (IANA).

But while the academic and professional research that CACM showcased was renowned and revered, the publication was also able to serve unofficially as a watercooler, a paper version of a newsgroup, not just programmers trying to figure out language or scholars debating standards, but in real-world issues. Through the years, CACM held a forum for computer folks who were looking for advice on how to bridge the divide between researchers in the labs and those who were implementing programming, and the other bridge between the developers and the end users.

There are so many thought-provoking pieces in CACM history, articles that looked at the big picture of a changing world, even as the images were still unfolding. In the early 1980s president David Brandin asked if our society will be vulnerable as a result of our growing dependence on computers and communication systems, an issue that is as relevant today as ever. In an exchange in "Letters to the Editor" decades before, the fragile and controversial sides of developing artificial intelligence are exposed and shine a light on the same issues the technology community, and culture at large, will have to address going forward.

There are broader questions raised in these pages too, ones that have more to do with conscience than science. For instance, Maurice Wilkes, in 1996, provides the argument for moving toward a more diverse, non-U.S.-centric international Internet. And in "The Net Progress and Opportunity," Larry Press implores the spread of the Internet to less wealthy nations. "As a major professional society ACM should also consider its role," he wrote in 1992.

CACM has been able to capture and present both science and commentary in a thought-provoking, educational way for 50 years. What has made the organization and the publication so enduring is its promotion of the free exchange of ideas. So Happy Anniversary, CACM. And congratulations on providing a forum where an ever-broadening community can discuss ideas that will ripple into yet more business sectors as the years go by. C

Meg McGinity Shannon (megshan98@yahoo.com) is a technology writer based on Long Island, NY.

CALL FOR PARTICIPATION

CTS 2008

Irvine, California, USA

The 2008 International Symposium on Collaborative Technologies and Systems

May 19 – 23, 2008
The Hyatt Regency Irvine Hotel
Irvine, California, USA

Important Dates:

Paper Submission Deadline	**January 7, 2008**
Workshop/Special Session Proposal Deadline	**December 14, 2007**
Tutorial/Demo/Panel Proposal Deadline	**December 21, 2007**
Notification of Acceptance	**February 8, 2008**
Final Papers Due	**March 3, 2008**

Conference Co-Chairs:

Bill McQuay, Air Force Research Laboratory, Wright Patterson AFB, USA
Waleed W. Smari, University of Dayton, USA

For more information, visit the CTS 2008 web site at:
http://cisedu.us/cis/cts/08/

In cooperation with the ACM, IEEE, IFIP

Peter A. Freeman

Back to Experimentation

Three forward-looking projects depend on experimentation under real-world conditions.

Some of us in the computing field have been around long enough to start to see the adage "History repeats itself" come true in the way we produce major advances in computing. Here, I want to note the recent emergence of serious experimentation on a scale not seen in years[1] and relate it to what some of us participated in when the field was young.

Since the late 1990s, a number of proposals and initial attempts have sought to develop and experiment with new technology under real-world, but nonproduction, conditions, on a scale relative[2] to the desired result not seen since CACM was young. Three such efforts point the way toward a renewed and very valuable trend of experimentation.

The most visible is the Defense Advanced Research Projects Agency-supported effort to build and operate robotic vehicles capable of driving themselves under demanding, real-world conditions. It has succeeded, not only operationally, but also in engaging the effort and imaginations of hundreds, perhaps thousands, of researchers and students, as well as the general public (en.wikipedia.org/wiki/ Darpa_grand_challenge). It is for roboticists to evaluate the technical results, but from my perspective it has been a great success in helping us all set our sights on what can be achieved through experimentation at scale.

The second, just starting to do some preliminary prototyping after extensive planning, is the Global Environment for Network Innovations Project (www.geni.net) begun in 2004 by the National Science Foundation's Directorate for Computer & Information Science & Engineering. GENI intends to refocus networking research on new architectures and mechanisms for future networks, not just on developing patches for our current networks. The project's Web site, which describes GENI and provides pointers to related information, is maintained by the GENI Project Office operated by BBN Technologies under agreement with NSF. GENI will support such research with a large-scale, experimental network that will be the largest experimental piece of "equipment" built solely for computer science research. It is not yet well known outside the computing research community, though such mainstream publications as *The New York Times* and *The Economist* have covered its progress. Meanwhile, it has already spurred networking and related research (including computer science theory and communications theory) and major responses from Europe (www.future-internet.eu/) and Japan (seen only in news reports at the time of this writing[3]).

The third effort—called by some "data-intensive supercomputing"—is still largely at the talking stage though appears to be gaining momentum (reportsarchive.adm.cs.cmu.edu/anon/2007/abstracts/07-128.html). Based on the idea that the massive,

[1]Despite serious experimentation in computing research (reflected in the special section "Experimental Computer Science," November 2007), from my perspective as a professor, we have not insisted on enough experimentation.

[2]"Relative" is the operant idea here. Most experimentation so far has been only a fraction of what a "fieldable" product or system might be, thus leaving open the question of scalability. One might argue, only slightly gratuitously, that some large government projects have indeed been "experiments"; unfortunately, they are rarely intended to be experiments, nor is much learned from the attempt in many cases.

[3]The Japanese Minister of Technology was widely quoted last summer (www.newlaunches.com/archives_japan_working_to_replace_the_internet.php), though he left office soon thereafter; plans are still being prepared.

LISA HANEY

constantly changing databases we all access (think Google) represent a new mode of computing and deserves to be explored more systematically. Various ideas are being developed on how to do this without becoming entangled in critical production processes.

They present great opportunities for advancing computer science and the technologies it makes possible. They also potentially involve extensive research activities, as well as significant investment in research infrastructure to enable the actual research. NSF spends 25%–30% of its annual budget on instruments to advance science in other fields, but computer science has not envisioned such large projects until recently.[4]

These observations led the CISE Directorate to issue a call for proposals to create a "community proxy responsible for facilitating the conceptualization and design of promising infrastructure-intensive projects identified by the computing research community to address compelling scientific 'grand challenges' in computing." In September 2006, NSF chose the Computing Research Association to create the Computing Community Consortium (www.cra.org/ccc/); now in operation, it looks to engage as many people and institutions in the research, education, and industrial communities as possible to fulfill its charter. At the heart of the effort is the understanding that major experimentation can and should be done in many cases before more expensive development and deployment are undertaken, something that industry alone can't afford to do.

All three efforts described here involve research characterized by observation, measurement, and analysis of results. While the same can be said of many industrial prototyping efforts and should also be true of small-scale academic research (such as thesis work), they are either impossible to do under large-scale, real-world conditions (in the case of academic research) or aren't done at all due to the pressure to produce near-term, profitable results. It's rare for experimentation to advance the boundaries of what we know how to do in computer science on a scale that is large relative to the state of the art.

The "relativity" factor has all but eliminated the kind of experimentation we did in the 1950s and 1960s. For example, in the mid-1960s, I was able and encouraged to build a small (four-user) time-sharing system on a minicomputer as a master's thesis that others could use in a production environment to see how well it worked and how it might change operations [1]. Even though it was tiny by today's standards, it was large relative to what existed then. I was able to do it because there were no such commercial systems then, and users were hungry for any improvement, even if it crashed some of the time. Today, it is impossible to mount a similar operating systems project, relative to what is required technically and expected by users.

This brings me back to the title of this column. The projects I've described here and the efforts to develop others portend the return of experimentation, somewhat in the style of the early days of computer science but with some important differences. First, while we should and indeed will see much more serious experimentation in the future, it will certainly be more costly than its counterparts years ago. Second, in some projects—perhaps most, given the practical nature of computing—experimenters must find ways to involve significant numbers of users in the "experiment"; this is a key feature of the GENI project. Third, and most important, they must employ much more careful observation, measurement, and analysis than was necessary or possible 50 years ago. So, I hope history really is repeating itself but this time improving what we do, how we do it, and the results all at the same time. C

Reference

1. Freeman, P. *Design Considerations for Time-Sharing Systems on Small Computers.* Master's Thesis, University of Texas at Austin, 1965.

Peter A. Freeman (freeman@cc.gatech.edu) is Emeritus Dean and Professor at Georgia Tech, Atlanta, GA. As Assistant Director of the National Science Foundation Directorate for Computer & Information Science & Engineering, 2002–2007, he was involved in starting the GENI project and the Computing Community Consortium.

[4]EarthScope (www.earthscope.org/) is an excellent example of how science and technology advances in other fields.

Fifty Years and Still Growing

This anniversary issue represents how *Communications* has covered the depth and breadth of the computing field over the decades, as well as its growth, and the industry's shifts in technological focus. The name "Association for Computing Machinery" once explained the primary interest of our members. Of course, over the last 50 years, both ACM and CACM have embraced systems, software, services, policy, the role professionals play, and other topics relating to information technology.

Subjects such as algorithms, architecture, operating systems, programming languages, networking, databases, software engineering, and artificial intelligence are at the heart of the "classic" CACM and have driven decades of progress. CACM published many great papers in these areas that are still quoted today.

Nevertheless, those subjects do not cover the bulk of activity in computing. Far more effort goes into design, integration, deployment, and support of applications than into the creation of new algorithms and core components. Successful software developers will always need to engineer core software, essential tools, and critical applications. But they must also apply their skills and knowledge to problems of societal, scientific, and commercial importance. In recent years, CACM has reflected these broader concerns.

The global IT industry will continue to expand, with turns toward services, integration, distributed computing, dynamic information, and increasing demands for reliability, security, usability, and accessibility. Research and advanced technology are growing in these areas, and are new generating true excitement and new possibilities.

ACM is creating a new editorial model for CACM (see Moshe Vardi's article, page 44). We have taken great care to ensure this plan reflects the needs of our readers, particularly their strong interest in areas at the forefront of the computing field—what is coming from research labs and appearing in advanced applications as well as what is required to deliver advanced systems. Our members are also keen to learn how government and policy initiatives can shape progress around the world.

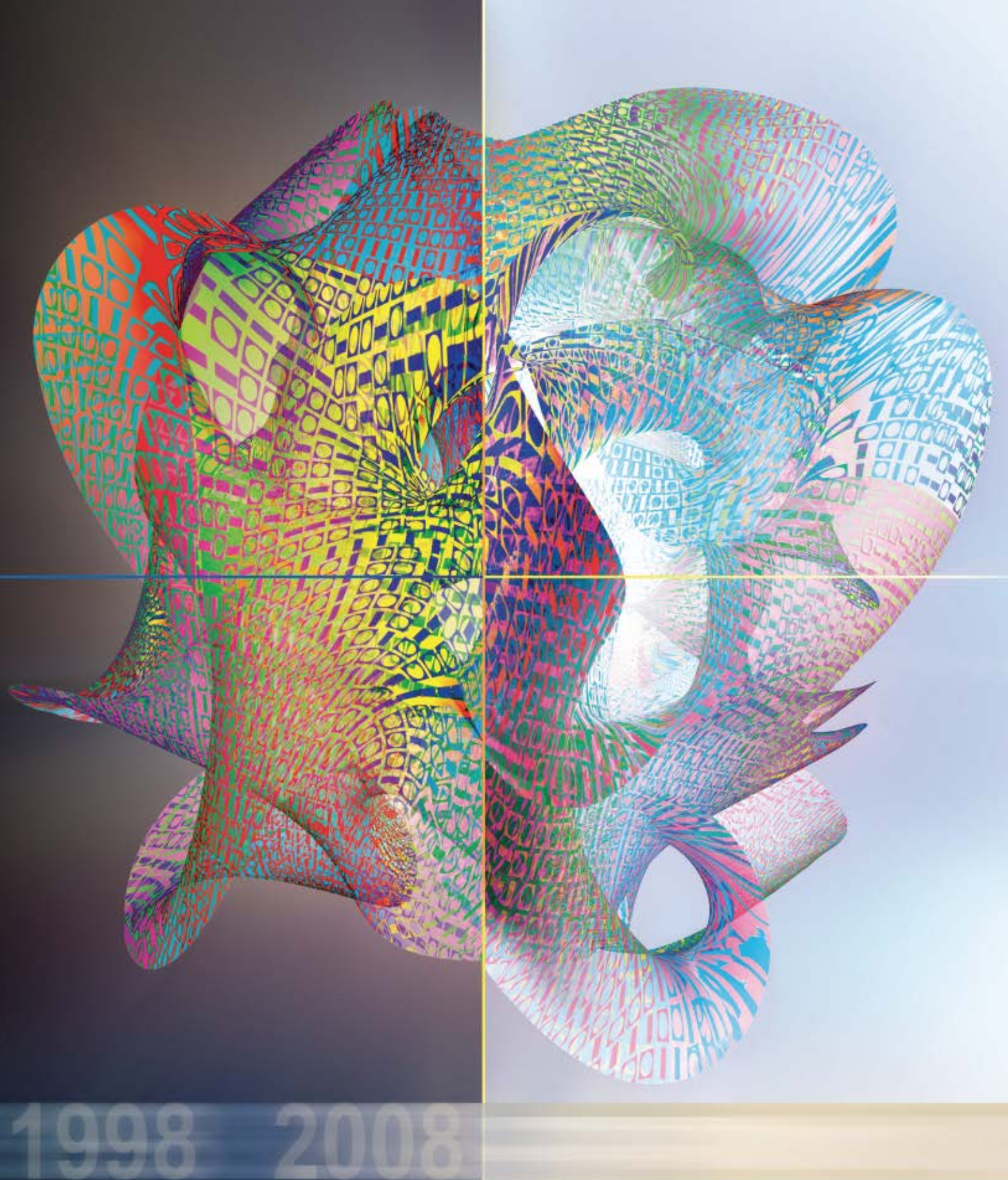
This issue sums up the glorious past of CACM. Its future should be even better! C

Stuart I. Feldman (sif@acm.org) is President of the ACM and Vice President–Engineering at Google.

FROM THE EDITOR'S DESK

Communications of the ACM has been most fortunate to have had seven authoritative stewards lead its editorial direction over most of these last 50 years. The editors-in-chief (EICs) who have served this publication, and the computer science community, with their foresight and dedication to the cause are some of the prime reasons why CACM continues to thrive even as computing interests, membership dynamics, and professional demands change and expand in scope. CACM's EICs were always focused on finding the common thread among a very broad-based audience, searching out the field's shining stars, and making sure the final product was always of the utmost quality and professional value.

In the following section, we present stories from the five surviving CACM EICs (**Calvin C. Gotlieb, M. Stuart Lynn, Robert L. Ashenhurst, Peter J. Denning, and Jacques Cohen**) who share some poignant memories from their days at the helm, as well as some of the challenges they faced, and their hopes for this publication as it moves forward. We also excerpt some of the decisive and intuitive editorial convictions of **Alan Perlis** (CACM's first EIC) and **Gerard Salton** (its third) as they embarked on their stellar editorships. And we introduce the newly appointed CACM EIC, **Moshe Y. Vardi**, who unveils news of an exciting editorial revitalization for this publication and the work that led to the creation of this plan.


1998 2008

ALAN J. PERLIS

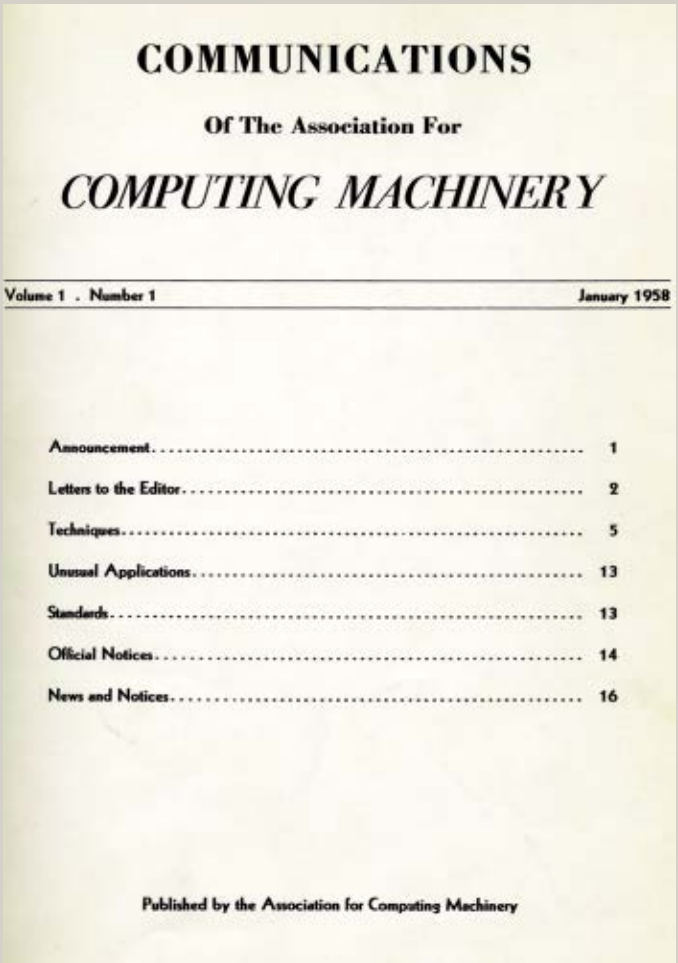
EIC YEARS JANUARY 1958–JULY 1962

AT FIRST GLANCE

Excerpted from Communications, *Jan. 1958, Volume 1, Issue 1, page 1.*

"The *Communications of the Association for Computing Machinery*, published by the Association for Computing Machinery, is a new monthly periodical, which will be mailed to all members of the Association and subscribers of the Journal at no added cost. This new periodical is intended primarily for the rapid dissemination of information whose kind and quality will be of value to the members of the Association.

...[T]his journal will provide space not elsewhere available for publishing worthwhile, but possibly fragmentary, developments in the use and understanding of computers, e.g., descriptions of computer programs, computer-inspired techniques in numerical analysis, and educational efforts, to name a few. The *Communications* will also provide a forum for the Association's membership with the Letters-to-the-Editor department. This department will be ideal for nourishing controversies that illuminate informed differences of opinion." C



COMMUNICATIONS

Of The Association For

COMPUTING MACHINERY

Volume 1 . Number 1 January 1958

Published by the Association for Computing Machinery

THE SIX-PERSON EDITORIAL TEAM FOR THE FIRST EDITION WERE EMPLOYED AT BURROUGHS CORP., RAMO-WOOLDRIDGE CORP., IBM, SPERRY RAND CORP., SYLVANIA ELECTRIC, AND CARNEGIE INSTITUTE OF TECHNOLOGY (PERLIS)

BY CALVIN C. GOTLIEB

EIC YEARS AUGUST 1962–DECEMBER 1965

A TIME TO RETROSPECT AND PROSPECT

As noted in these introductory pages, Alan Perlis was the founding Editor-in-Chief (EIC) of *Communications*, with the first issue debuting in January 1958. He resigned upon being elected ACM President in June 1962. During his tenure, CACM content was organized into departments, such as "Scientific Applications," "Standards," "Programming Languages," among others. I was the editor of a section on "Business Applications."

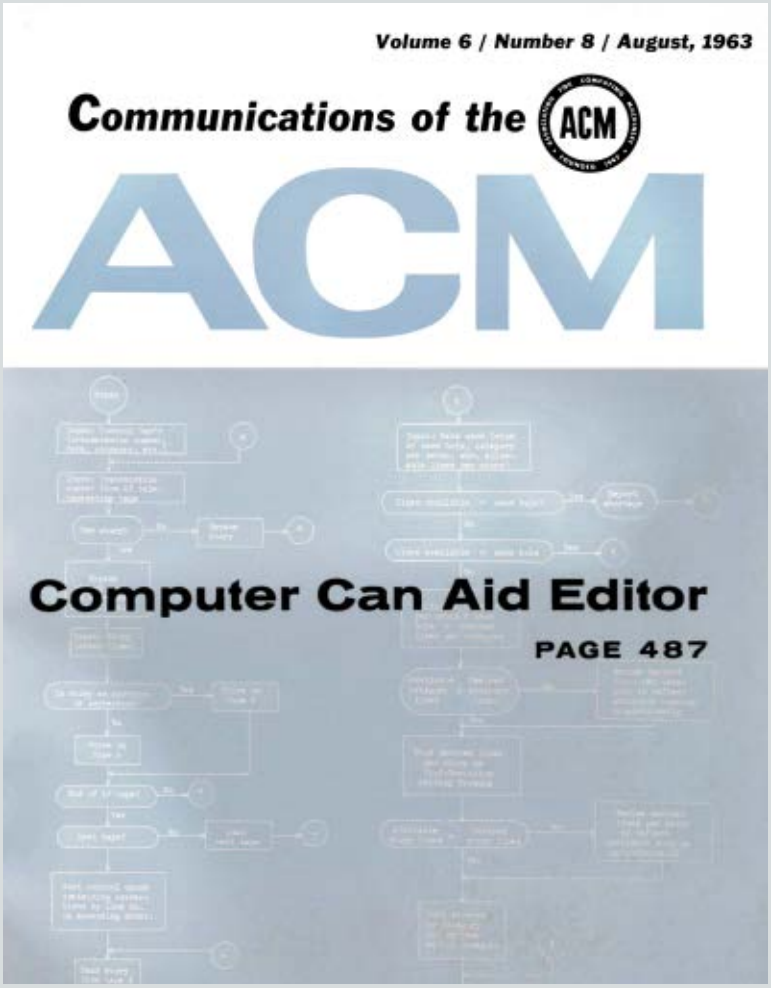
I accepted an invitation to become the EIC of CACM, starting with the August 1962 issue (Vol. 5, Issue 8), and continued in that position until the December 1965 issue (Vol. 8, Issue 12), at which time I assumed the editorship of *Journal of the ACM* (JACM). I was succeeded at CACM by Gerry Salton.

My contribution to the 50th anniversary issue of JACM was entitled "A Golden Time for JACM" where I noted that numerous authors who were already famous for their work with computers, or destined to become so, authored papers in JACM during my editorship. The same can certainly be said for CACM.

In the 41 issues during my CACM tenure, there were no fewer than 10 individuals who would later win ACM's A.M. Turing Award, five who would become ACM Presidents (see the sidebar on page 40), and numerous European contributors who were the principal researchers on computers and applications in their own countries (among these latter were Henri Rutishauer, Peter Henrici, Niklaus Wirth, Fritz Bauer, A. Winjgaarden, and Edgsar Dijkstra). In many cases there were multiple contributions from those mentioned, and a full list of CACM authors in this brief period is a good beginning to a list of computing pioneers worldwide.

The period was one in which there was tremendous activity around computers on many fronts. Saul Gorn noted that ACM membership had reached 13,000 and gave reasons why he was convinced that computer science was certain to assume major importance. An ACM committee consisting of Sam

"In the 41 issues during my CACM tenure, there were no fewer than 10 individuals who would later win ACM's A.M. Turing Award, five who would become ACM Presidents.



THE COVER STORY IN THE AUGUST 1963 ISSUE PRESENTED A COMPUTER PROGRAM DESIGNED TO EDIT NEWS STORIES IN A NEWSPAPER STYLE.

Conte, John Hamblin, David Young, Werner Rheinbolt, and others produced preliminary recommendations for an undergraduate program in computer science. Joint Computer Conferences, co-sponsored with the IEEE-Computer Society, were held semi-annually, but questions had already surfaced regarding the appropriateness of ACM getting actively involved in hardware exhibitions. Under the leadership of Isaac Auerbach, the International Federation of Information Processing Societies (IFIP) was gaining worldwide recognition.

In 1960 Algol 60 had appeared as the result of an international collaboration, and with its many elegant features, including a highly structured format, and recursive subroutines, proved that a programming language designed by a committee did not have to be bureaucratic or convoluted.

A revised version of the Algol 60 Report was published in the January 1963 issue of CACM, and in the various issues of that period, there were many so-called "Certifications" of Algol algorithms that were a consequence of the universal interest in the language.

From time to time throughout my CACM editorship and afterward I contributed my own work and opinions. In the April 1969 issue, (when Stuart Lynn was EIC), in a piece entitled "On the ACM Publications," I reported on the conclusions of an ad hoc committee of the Editorial Board established to respond to a request for the ACM President to formulate a five-year policy for Council consideration. On rereading that report I am struck by the extent to which issues discussed there continue to demand attention to this very day when the shape of CACM is being recast.

One question, which certainly survives, was how to make better use of technology in ACM's publications. In those days technology was directed to automatic indexing and abstraction, selective dissemination of information, and to the publishing process itself. There was a strong feeling that ACM was not taking sufficient advantage of the expertise of its membership. Gerry Salton, in "Towards a Publication Policy for ACM," (CACM, Jan. 1966, p. 2), presented a strong case for making it possible to submit papers in machine-readable form. Today it is a matter of coming to terms with the Internet, blogs, and other online dissemination of facts and opinions.

One question that was debated, but for which there is now a different answer than originally given, was the necessity for articles to be robustly refereed. CACM's Editorial Board took the position (which

I supported) that it was important to continue placing major emphasis on refereeing. The reason given was that many universities were still in the process of establishing formal computer science programs and the presence of a highly respected literature, particularly in JACM and CACM, was an important factor in having these programs accepted. Indeed, three of the courses noted in the 1968 report by ACM's Curriculum Committee on Computer Science, listed JACM or CACM in 38 of 78 references (49%).

Today, with a majority of the membership coming from practitioners rather than academics, this argument no longer holds the same force.

This anniversary issue offers a time for retrospect and a time for prospect. Retrospect means remembering all the colleagues, friends, and excitement of those heady days. As for prospects, anyone who reads Moshe Vardi's account of the comprehensive research and imaginative thought that has driven the deliberations about the future course for CACM (see p. 44), will know the publication is in very good hands indeed.

Calvin C. (Kelly) Gotlieb
(ccg@cs.toronto.edu) is currently Professor Emeritus in Computer Science at the University of Toronto.

GERARD SALTON

EIC YEARS JANUARY 1966–DECEMBER 1968

THE POWER OF PRINT

Excerpted from "Toward a Publications Policy for ACM," Communications, *Jan. 1966.*

"With the present issue I am taking on a new responsibility as editor of the *Communications of the ACM*. I hope that my inexperience, when compared with the impressive performance of my predecessors, will not become too obvious to most of you, and that you will continue to consider the *Communications* as a primary source for learning what is new in the field and as an interesting opportunity for the publication and dissemination of your own work.

... One of the encouraging thoughts for an incoming editor of a technical journal is the often heard assertion that editors may shortly see their burdens considerably lightened. Indeed, it is said that the job of a technical editor may one day be abolished altogether: in an era in which consoles may soon be found in every bedroom, technical journals and their editors may be replaced by a system of universal, personalized dissemination of information, and editors will, I suspect, be the first to welcome such a development. But this day is not yet, and my own feeling is that some form of printed record of technical material is likely to remain with us for a long time to come.

Even if an editor is thus temporarily kept from premature retirement, it is still necessary for him to face the question whether it is safe, or appropriate to relax, and to let things go on as heretofore, or whether on the contrary, he should take notice of those developments in automatic information handling which are likely to affect the publishing process. Such a question is particularly germane in the ACM context, since the most important changes in information dissemination and retrieval are directly caused by certain novel uses of computers."

BY M. STUART LYNN

EIC YEARS JANUARY 1969–MARCH 1973

THE BATTLE OF THE COVERS

The battle of the covers. That was the beginning.

I had just taken over as Editor-in-Chief of CACM in 1969. Don Madden was then ACM's Executive Director. With undoubtedly some justification, he was concerned that the covers of CACM were too drab and were not attractive to an increasingly broad base of membership. He wanted to brighten them up.

Gerry Salton was my predecessor as EIC but had moved over to edit *Journal of the ACM* (JACM). He had suggested me as his replacement (I had been editor of CACM's then "Scientific Applications" department). I felt it was my duty—following in Gerry's distinguished footsteps—to protect the prerogatives of the volunteer EIC as *El Supremo* and that I, not Don, should be making the decisions about CACM covers. After all, CACM was primarily a prestigious research journal (with ACM news thrown in), not some fly-by-night computer magazine.

Kelly Gottlieb chaired the Editorial Board, at the time the governing body for publications. He and the Board tried to referee this mother of all publications battles. Both Don and I were adamant in our positions.

Kelly and the Editorial Board found the right solution. They settled on the notion (later endorsed by Council) that there should be an oversight Publications Board composed of broader representation—not just editors—who would bring both the business and the editorial perspective into consideration. That was the beginning of a process that eventually transformed ACM's publications.

As a small palliative step, we did change CACM's covers a bit during my editorship, attempting to bring some cohesion into the look and feel. The covers were attractively (but not garishly) designed, all in black and blue colors (see the accompanying examples). They were certainly more pleasing, but surely not as much as Don would have wanted.

In spite of the introduction of some materials of broader interest such as Forum (edited by Bob

Ashenhurst), the essence of CACM did not change that much under my editorship. It was still predominantly a refereed research publication—a tradition that was significant to the research community who strongly resisted any change to widen the appeal of CACM.

The growing and broader membership of ACM was, however, not happy. They did not see why they should receive each month for their dues a publication that many could barely understand if at all. The Publications Board (by then I had taken over as Chair; while Bob Ashenhurst ably took over as CACM EIC) felt that a more strategic approach was needed to set the future direction for ACM's publications.

We formed a committee, the refuge of all desperate chairs—the Publications Planning Committee (PPC) with representation from many different constituencies.[1] The PPC took a top-down approach. First step: obtain broad agreement on the strategic purposes of ACM's publications. Second step: fill in the details. Covers were not the issue!

After months of tortuous deliberation, the PPC produced the Long-Range Conceptual Framework for ACM Publications (quite a mouthful!), or LRCF, as a broad policy document. Fundamental to the policy

(A)

(B)

(A) JANUARY 1971 (BEFORE);
(B) FEBRUARY 1971 (AFTER)

[1]I served as Chair of the committee, which included Bob Ashenhurst, Dick Canning, Ray Miller, Christine Montgomery, Joel Moses, Tom Murray, Jean Sammet, and Evelyn Swan.

vision was the recognition that ACM needed to create a framework in which many research publications could flourish to reflect the burgeoning of computer science (*Transactions on Mathematical Software* and *Transactions on Database Systems* had already been launched), and that more flexibility was needed if this was to happen. In fact, CACM as a purely research publication was ironically inhibiting this growth, sapping some of the best papers that could instead be used as a critical mass to launch new research publications. At the same time there was a growing backlog in CACM and the ensuing publication delays understandably upset many authors. We proposed a structure that we believed would lead to the birth of many more research Transactions and also shorten publication lead times.

To complete this picture, conversely, CACM had to change radically. Research articles would be moved to JACM and to existing and new Transactions. In their stead, CACM would publish articles of broader appeal that would nevertheless be authoritative and definitive. The revised CACM was, as Peter Denning writes in his essay, conceptually called "JAM"—the Journal for All Members. We also developed a somewhat pedantic taxonomy for the Transactions that, correctly, has long since been abandoned.

After considerable debate, head scratching, and socialization, the Publications Board and Council adopted this new "Conceptual Framework." There was much opposition, of course, from many researchers who (at least initially) felt they were being shortchanged, losing the opportunity for their published papers to reach an audience of tens of thousands, regardless of whether the majority of those "readers" would ever look at those papers. Others felt that some of the best papers would escape ACM's publications altogether and be published elsewhere—a legitimate concern at the time, even though it did not in fact happen. Yet others felt that the proposed directions for CACM did not reach far enough into transforming it into a competitive commercial product. Undoubtedly, there are still many today who do not accept the "CACM decision."

We went forward. Peter Denning took the helm of the 'new' CACM and did a brilliant job of transforming concept into reality. CACM did not descend into the throwaway rag that many feared. The contents became much more readable by the broader membership, but the articles were still important and definitive, providing deeper understanding of trends and issues in the field. We launched many new Transactions, significantly increasing opportunities for the publication of research material. Each new publication stood on its own feet financially.

"[W]ith this issue we are initiating several approaches that represent new directions. This is so that we may better serve the needs of the Association both in terms of the requirements of our profession and in terms of the diverse interests of our readers."

—M. Stuart Lynn, CACM EIC, February 1971.

I am long retired and out of touch with the field—and with ACM. My association with ACM's publications and Council ended in the early 1980s. I now see computing from the perspective of an avid user who benefits every day from concepts, ideas, and developments that were first nurtured in ACM's publications. I feel a sense of pride that I in my own small way—along with so many great colleagues and a terrific staff—was involved as a catalyst in making some of this possible.

I recently browsed ACM's publications list. It is extraordinary! The changes, of course, have been enormous even looking beyond the transition to so much material available online—a direction not anticipated in the LRCF. There are now over 30 Transactions, six journals, and numerous other publications far extending the early four publications of JACM (the very first), CACM, *Computing Reviews* and *Computing Surveys* all of which still flourish. Most of this explosive growth occurred long after my years of association with ACM.

But it all began with the battle of the covers! C

M. Stuart Lynn (mslynn@mac.com) is retired and living in Palm Springs, CA.

BY ROBERT L. ASHENHURST

EIC YEARS APRIL 1973–JANUARY 1983

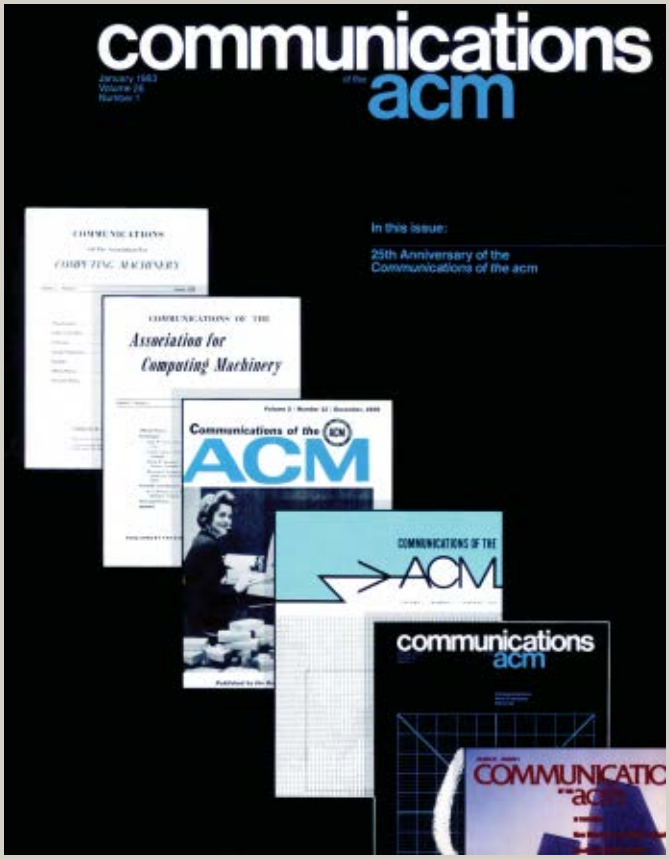
THE BATTLE BEHIND THE SCENES

Like Stuart Lynn, I had been a department editor for CACM before becoming its EIC. In fact, the "Computer Systems" department was initiated by Kelly Gotlieb during his editorship. He sent me a letter enclosing a submitted paper that he thought should be published in CACM; however, it fit no existing department. It would appear in the September 1965 issue: Reilly and Federighi's "On Reversible Subroutines and Computers that Run Backwards." A subsequent contribution, of considerably more lasting significance, was published in that same issue: E.W. Dijkstra's "Solution of a Problem in Concurrent Program Control." Indeed, the paper initiated a whole new subdiscipline in the computer operating systems area. Eventually, "Computer Systems" became the department for software/hardware systems papers such as those that now appear in *Transactions on Computer Systems (TOCS).*

Stuart has chosen to call his reminiscence "The Battle of the Covers." Mine is appropriately titled "The Battle Behind the Scenes" (I eschew the expression "Under the Covers," used by Ted Codd to describe relational database infrastructure, since CACM is a family publication, sort of). Researchers in programming languages and other disciplines liked CACM as it was—a vehicle for presenting their research results, one that was appropriately refereed but also widely read (or at least widely circulated), as the publication was distributed to all ACM members. At the same time, practitioners in computing felt its articles were arcane and basically unreadable; they resented having to receive it.

During that time, several Transactions were created as spin-offs for researchers in specific domains—first *Transactions on Mathematical Software (TOMS)*, then *Transactions on Database Systems (TODS), Transactions on Programming Languages and Systems (TOPLAS), Transactions on Computer Systems (TOCS)*, and many others. These publications attracted papers in some of the central research areas in computing, even without the wide circulation of CACM, and it was felt the departments that remained would not be sufficiently focused to support the less-central research topics. Thus, the ACM Publications Board proceeded apace with a plan for a "new" CACM as described briefly by Stuart (to avoid stepping on any toes, let me call this the "old new" CACM to contrast it with the various "newer new" versions generated subsequently).

While this long-range solution was being formulated, however, we mounted a number of short-range efforts to address some of the discontent among ACM's practitioner members. We added a specially edited "Computing Practices" section, with contributions reviewed, but not refereed, by research standards. As noted by my fellow editors, we used the acronym JAM—Journal for All Members—in discussing these matters. I



THE 25TH ANNIVERSARY ISSUE OF *COMMUNICATIONS*, PUBLISHED IN JANUARY 1983, CELEBRATED PAST EDITORIAL ACHIEVEMENTS BY REPRINTING 21 SEMINAL PAPERS FROM ITS FIRST QUARTER-CENTURY.

wrote a couple of pieces (in the July 1977 and Feb. 1982 issues) counting how many pages of "nonresearch" material actually appeared between the covers of CACM. In addition to the existing "Reports and Articles" sections, these included the calendar, position notices, and other materials of general member interest. I even suggested (facetiously) that those who could not bear the sight of the academic content physically tear out the offending pages, traditionally grouped together in the middle of the issue. Left would be virtual JAM (or vJAM).

Eventually, of course, the last of the "black-and-blue" covers appeared with a colorful hint of what was to come peeping out of the bottom right-hand corner of the cover of the 25th Anniversary issue in January 1983. This edition, my swan song as EIC, reprinted 21 notable papers of the past, including Dijkstra's previously mentioned piece, as well as E.F. Codd's milestone "A Relational Model of Data for Large Shared Data Banks" (June 1970), and the seminal "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," by R.L. Rivest, A. Shamir, and L. Adleman (Feb. 1978). Also in that issue was a page on CACM's contributions to computing education, as well as a summary of its history entitled "The First Quarter Century," complete with timeline.

The "Forum" section, consisting of letters to the editor contributed by readers, began before my editorship, but grew and flourished mightily during my years. We spun off "Technical Correspondence" as a separate section, so that Forum could be a source of lively opinion on all things related to computing, however remotely. When I took the helm, being Forum Editor was an implied adjunct to being EIC. When I left the latter post in 1983, I remained as Forum Editor until 1991. Editing Forum was definitely one of the most stimulating and fun parts of the job, and I tried conscientiously to print everything that came in, no matter how scurrilous it appeared to many. I also endeavored to get responses appearing in the same issue from those criticized (or maligned). The only letters "suppressed," by order of the ACM Council, were those supposedly of general technical interest but submitted by members who were currently candidates for ACM elected office (thus branded "electioneering"). I particularly remember a letter from Herb Grosch, perennial gadfly, accompanied by a second letter—of protest—in case we declined to print the first one.

From these roots grew the galaxy of ACM publications we have today, covering all aspects of computing for all manner of professionals. One might think, of course, that this was inevitable, given the pervasive growth of the computing field. But it certainly owes a good deal to the vision and foresight of all those involved in ACM publications in the earlier years. C

ROBERT L. ASHENHURST (frashen2@chicagogsb.edu) is Professor Emeritus at the University of Chicago, IL.

BY PETER J. DENNING

EIC YEARS FEBRUARY 1983–SEPTEMBER 1992

DÉJÀ VU ALL OVER AGAIN

After a 10-year struggle within ACM to define a Journal for All Members (JAM), a "new" *Communications* was launched in the cold of February 1983. CACM was to leave behind its pure research past and transform into a professionally useful, interesting, monthly magazine for all members. The CACM that evolved in the decade following 1983 is substantially the form you find today. I was the EIC who managed the transition.

To understand the "new" CACM, you need to understand the "old" CACM that preceded it. Stu Lynn has reported that a simple disagreement over the covers led to the formation of the ACM Publications Board in the mid-1970s and to a major restructuring of the ACM publications in the late 1970s. The 1970s were a major growth phase for ACM and the computing field, with a continuous stream of amazing new discoveries and inventions. ACM offered its authors two research publishing venues: *Journal of the ACM* (JACM) and CACM. CACM was the preference for papers about systems, architectures, and applications; JACM for theoretical papers. But these two journals could not accommodate the growth of the computing field.

THE PUBLICATIONS STRUGGLE

By the 1970s, the publications budget, which covered JACM, *Computing Reviews, Computing Surveys*, and CACM, was about half the ACM budget. The member cost of CACM alone was about half the annual dues. ACM revenues were very tight and everyone was sensitive about returns on investment.

CACM and JACM could not keep up with the explosive growth of scientific discoveries and technology inventions. By the mid-1970s there were major queues—and delays averaging three years—in both publications. Authors and readers alike complained bitterly to the ACM leadership and Council. Presi-

dential campaigns turned on proposals for improving publications, especially CACM. Unfortunately, there was not enough money to pay for the additional pages that would eliminate the CACM backlog. And even if there were, a typical issue would be over half an inch thick! Eventually there was a consensus favoring a major restructuring of publications to allow for more research publications, each self-supporting with its own subscriber base.

At the same time, an increasing number of SIGs wanted to start Transactions in their disciplines. The most active promoters were programming languages, computing systems, databases, graphics, and office automation. The SIGs had surplus funds to put into these publications.

Under the leadership of President Tony Ralston, ACM backed a project in AFIPS (American Federation for Information Processing Societies, now defunct) to launch a *Scientific American*-style magazine for computing. It was called *Abacus*. The prototypes were slick and compelling. Around 1975, AFIPS declined to launch *Abacus* for financial reasons. Ralston tried to persuade ACM to launch a scaled-down version of *Abacus*, but it was too costly. The *Abacus* concept, however, established a beachhead in the minds of everyone thinking about the form of an improved CACM.

In 1978, the Publications Board, under the leadership of Stu Lynn, forged a consensus around a long-range publications plan. The plan called for the establishment of a line of self-supporting research Transactions in areas of established need. New Transactions in the areas of greatest backlog in CACM were of highest priority. By 1983, six Transactions were launched and the corresponding departments discontinued in CACM. Today, there are 32 Transactions and five more are on the way.

The long-range plan also called for CACM to transform into a concept called "Journal for All Members" that included aspects of *Abacus*. However, it took until 1982 for enough of a consensus to form around this idea that it could be incorporated into CACM.

Coming to a Head: 1982

When I was president of ACM (1980–1982) I heard numerous complaints about CACM. At that time, six Transactions had been launched or were about to debut, and CACM's corresponding research departments were eliminated. Although the backlogs were gone, so was the technical content. Now the readership had no news whatsoever about research advances in computer systems, databases, graphics, programming languages, or computer architecture. At least with the backlogs they saw three-year-old material. Now they saw nothing.

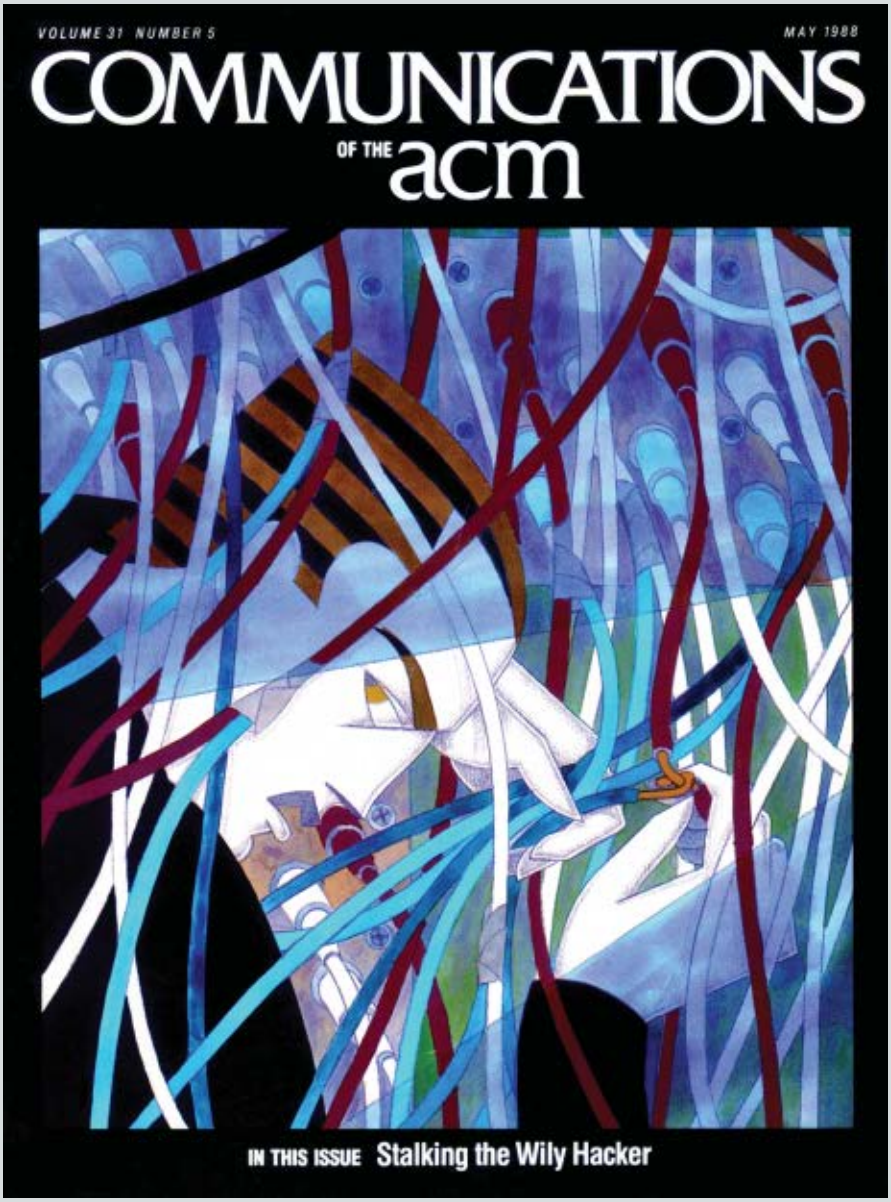
I spent a lot of time working with ACM leadership to forge a consensus around the JAM ideas as a way to transform CACM and respond to the members. The Council asked me to serve as EIC when the new CACM launched in early 1983. With the active participation of ACM Council, we put together a plan for CACM with these elements:

1. *News*. Refocus from ACM to industry. Eventually spin off all ACM news and calendars into a separate newsletter. (Done in 1990 with the debut of ACM MemberNet.)

2. *Computing Practices*. Expand coverage of technology topics, case studies, and how-to articles for practitioners, especially software developers. Hire new editors and writers to work proactively with practitioners to develop articles. (Ed Sibley was the chief editor for this.)

3. *Research*. Continue the existing research departments in emerging areas. Work with SIG conferences to get best papers in all other areas, especially in the departments that had been spun off to Transactions. Rewrite these articles so they can be appreciated by ACM professionals outside the immediate research area of the author. Where necessary, get experts to write opening perspectives to help readers appreciate the context and significance of a research paper.

To understand the "new" CACM, you need to understand the "old" CACM that has preceded it.



THE MAY 1988 ISSUE FEATURED A FIRST-PERSON ACCOUNT OF CLIFFORD STOLL'S YEAR-LONG ODYSSEY OF SILENTLY TRACKING A GERMAN COMPUTER PROGRAMMER WHO BROKE INTO THE COMPUTER SYSTEM AT LAWRENCE BERKELEY NATIONAL LABORATORY (AMONG 40 OTHERS WORLDWIDE) TO DISCOVER THE INTRUDER WAS A SPY SELLING SOFTWARE AND MILITARY DATA TO THE KGB. THE STORY, "STALKING THE WILY HACKER," RECEIVED WORLDWIDE MEDIA COVERAGE.

4. *Articles.* Establish a new line of contributed and professionally written articles in the *Abacus* style.

5. *Columns.* Commission regular columns from excellent writers. (The first was "Programming Pearls" by Jon Bentley, beginning August 1983.)

6. *Design.* Hire a professional design company to create a new look and feel for CACM that integrated all the elements noted here. Consult with them on every issue.

This plan drew on the many ideas from the JAM proposals, reader surveys, and comments. We believed it would establish a new balance among these elements that would prove to be much more satisfactory than the CACM of the day.

Council endorsed the final design and editorial plan in 1982. The new CACM was launched in February 1983 after a special issue in January to commemorate the best of CACM in its first 25 years.

But there was one problem: ACM Council wanted us to implement the plan but did not have the funds to hire all the staff required to execute the plan. We were able to hire two new editors and one journalist, but not the five editors and three journalists we thought we needed.

Therefore much of my time as EIC was spent on finding creative ways to implement as much

of the plan as possible within a meager budget.

What Worked and What Didn't

The news section took several years to find its footing. The biggest problem was finding news items that would still be fresh by the time the issue was published.

"Computing Practices" was our biggest challenge. Many practitioners are not inclined to publish and so it is necessary for the editorial staff to visit many conferences as well as solicit and write articles. We hired journalist Karen Frenkel, who wrote many articles and conducted many interviews; but these articles were quite labor-intensive. We needed three more Karens, but we did not have the budget. Her works were a big hit with readers. Once, Karen and I visited Apple Computer to interview Steve Jobs (published April 1989). When we asked if he thought the Internet would be crippled by hackers, he buried his head in his hands for a full minute; then looked up and said, "No, they see it as a critical infrastructure for their own work."

Another major success was the case studies conducted by Alfred Spector and David Gifford of MIT, who visited project managers and engineers at major companies and interviewed them about their projects, producing no-holds-barred pieces. This section was wildly popular among the readers. Unfortunately, the labor-intensive demands of the post got the best of them after three years, and we were not able to replace them. Also by that time, companies were getting more circumspect about discussing failures and lessons learned in public forums.

I would say we improved CACM's coverage of computing practices, but not to the degree we envisioned. In 2002, former ACM president Stephen Bourne persuaded Council to undertake a major initiative in the computing practices area by founding *Queue* magazine. *Queue* got the budget needed to do this right and ACM finally learned how to do it well.

Readable research. We found that many of the articles submitted to the remaining research departments were much less technical than articles submitted to the old departments. It was much easier to edit them into the article format. We also found that making arrangements with SIG conferences for best papers was much more difficult than we thought; they were not a fruitful source for CACM.

When we saw this approach to research was not viable, we seriously investigated imitating *Science* magazine's approach. The idea would be to invite research papers from all sectors of computing, edit the acceptable ones heavily to make them accessible to our audience, and have a rapid review process. We envisioned a day when the *New York Times* would cite a scientific breakthrough in a forthcoming article in the CACM—just like in *Science*. We visited *Science* magazine to find out how they do it. To our dismay, we discovered that the number of staff required to handle the rapid review and editing process was well beyond our means. We abandoned this idea.

Eventually we decided to discontinue the research category altogether and concentrate on doing the articles category well.

Articles. It was quickly apparent that our resources would not allow us to realize our dream of giving articles the full *Abacus* treatment. we would need 10 articles editors and we only had two. Moreover, we knew that many *Scientific American* readers found the articles shallow, and many authors felt their work was so rewritten it was no longer theirs. By 1985 we had abandoned the *Scientific American* model and settled instead on Sigma X's *American Scientist* model. Their editors solicit papers from leading researchers, asking them to write articles specifically for their publication. Editors work with authors to improve sentence and article structure for the best connection with the reader; the objective is to improve readability while retaining the author's own voice. *American Scientist* readers felt its articles had good depth, and authors felt it was still their own work. We could provide the editing and scouting needed to run this model from within our existing resources.

We established regular special sections to concentrate on emerging areas discovered by our editors. One of our first was a compendium of the best computing humor of all time (Apr. 1984, with Peter Neumann as editor). Our first outreach section—Computing in the Frontiers of Science—was published as a joint venture with the IEEE Computer Society (Nov. 1985).

Columns. We cultivated a stable of regular columnists to comment on a variety of issues. The first was Jon Bentley's "Program-

In the grand traditions of ACM, there are always people who think we can do a better job.

ming Pearls," (1983), which proved to be the CACM's most popular column of all time. After five years, Jon retired from the job, saying he was burned out from the schedule. "Literate Programming" in 1988 (Chris van Wyck), "Legally Speaking" in 1990 (Pamela Samuelson), "Inside Risks" in 1990 (Peter Neumann), and "Viewpoint" in 1983. Reader surveys told us this was the most popular feature in CACM; the majority of readers turned first to the columns section.

Design. The redesign was a complete overhaul: new typography, stylistic opening pages to articles, illustrations, and professionally designed covers. Our Fifth Generation Computing Systems cover won an award (Sept. 1983). In 1990, we moved all graphic design and layout in-house.

Mission Accomplished

We launched in 1983 with the mission given us by the ACM Council: Transform CACM to a magazine style, embodying the JAM concepts that would be interesting and useful to members every month.

We conducted regular reader surveys and focus groups to help us assess how well we were doing; and we made many adjustments. We continued to be very creative because the budget was not there to hire the personnel needed to fully realize the mission.

A number of our issues and covers received industry awards.

A recent survey of scientific journals confirmed that CACM is now highly ranked. It has the third-highest citation count across four key computing categories: Software Engineering, Information Systems, Hardware and Architecture, and Theory and Methods. As a result of this increased reputation, the submission rate for good articles has been rising.

We believe we achieved our mission and helped CACM achieve a high stature in the community.

I stepped down in 1992 to chair the Publications Board and lead the Digital Library Project.

Wisdom of the Ages

In the grand traditions of ACM, there are always people who think we can do a better job. When David Patterson was president of ACM, many researchers told him they thought CACM had never regained its vaunted glory of the 1970s. Patterson set up a committee to review the current model and propose ways to recharge its content and scope.

When I first talked with the committee, they were not aware that the reason many research departments had left CACM was the Publications Plan approved by Council in 1978. It was not the work of capricious editors, but of top ACM and SIG leadership.

Moshe Vardi was tapped to spirit this revitalization effort. He spent months gathering feedback from focus groups, studying reader surveys, talking with many individuals, and reviewing every aspect of CACM from bottom to top. A new CACM plan was proposed (see page 44).

It's the same plan we submitted in 1982! Right down to the models envisioned for each section. We thought our plan then—developed through a consensus process—was sound and I am delighted the consensus today is much the same.

There is one major difference. The current ACM leadership has agreed to fully fund the plan. They will be able to hire all the editors they need. No cutting corners. CACM can now become truly great. C

Peter J. Denning (pjd@nps.edu) is the director of the Cebrowski Institute for Innovation and Information Superiority in the Naval Postgraduate School in Monterey, CA.

ACM AUTHORS TURNED A.M. TURING WINNERS AND ACM PRESIDENTS*

THE GOTLIEB YEARS 1962–1964

Future Turing Recipients
Alan Perlis (1966)
Maurice Wilkes (1967)
John McCarthy (1971)
E.W. Dijkstra (1972)
Donald Knuth (1974)
Allen Newell (1975)
John Backus (1977)
William Kahan (1989)
Fernando Corbato (1990)
Peter Naur (2005)

Future ACM Presidents
Harry Huskey (1960–1962)
George Forsythe (1964–1966)
Anthony Oettinger (1966–1968)
Bernard Galler (1968–1970)
John White (1990–1992)

THE LYNN YEARS 1969–March 1973

Future Turing Recipients
Charles Bachman (1973)
Frederick P. Brooks (1999)
E.F. Codd (1981)
C.A.R. Hoare (1980)
Donald E. Knuth (1974)
Niklaus Wirth (1984)

THE ASHENHURST YEARS April 1973–1983

Future Turing Recipients
Butler Lampson (1992)
Dennis M. Ritchie (1983)
Ken Thompson (1983)
C.A.R. Hoare (1980)
Ronald Rivest (2002)
John Cocke (1987)
Francis Allen (2006)
Nicklaus Wirth (1984)
Adi Shamir (2002)
Leonard Adleman (2002)
Peter Naur (2005)
Robert W. Floyd (1978)
Richard M. Karp (1985)
Ed Feigenbaum (1994)

Future ACM Presidents
Stuart Zweben (1994–1996)
David Patterson (2004–2006)

THE DENNING YEARS 1983–1992

Future Turing Recipients
Robert E. Tarjan (1986)
Jim Gray (1998)
Ronald L. Rivest (2002)
Adi Shamir (2002)

Future ACM Presidents
Stuart Zweben (1994–1996)
Barbara Simons (1998–2000)
David Patterson (2004–2006)

THE COHEN YEARS 1992–1996

Future Turing Recipients
Juris Hartmanis (1993)
Frederick P. Brooks (1999)
Robert E. Kahn (2004)

Future ACM Presidents
Barbara Simons (1998–2000)
Maria Klawe (2002–2004)

*All works were published prior to award/office announcements.

BY JACQUES COHEN

EIC YEARS OCTOBER 1992–DECEMBER 1996

FROM ACADEMIA TO THE EDITORSHIP

Communications has always had a special meaning to me since the beginning of my career, both professionally and personally. My fascination with computers started in the late 1950s when I was pursuing my doctoral degree at the University of Illinois, Urbana-Champaign. The *Illiac-I* was among the early computers built in the U.S. and I had the privilege to use it extensively in my dissertation.

The experience with the *Illiac-I* changed my life. After undergoing the exhilaration of having *my* programmed instructions executed at lightning speed—it was milliseconds in those days!—I was convinced that computers would profoundly affect science and engineering. To me, it seemed mandatory to take part in helping propagate the use of computers.

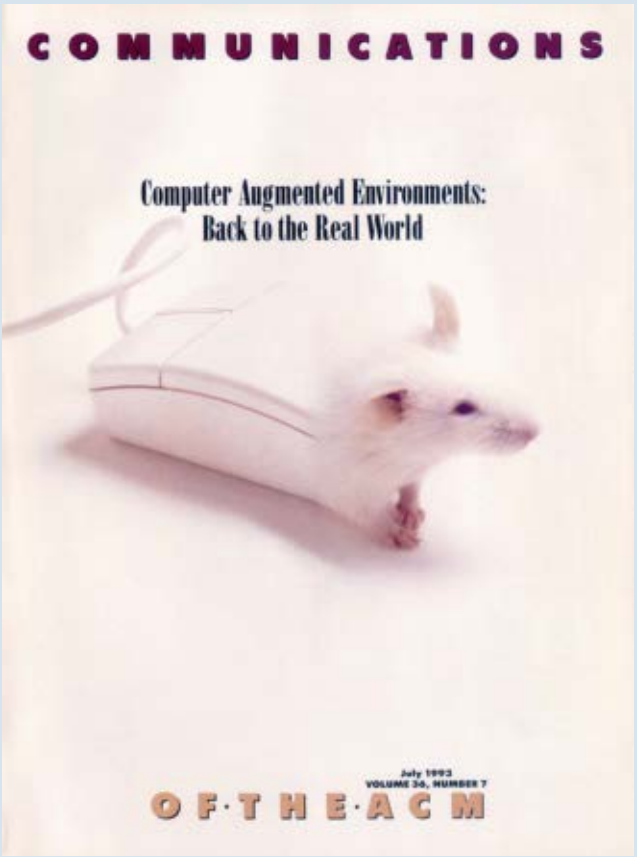
I became a member of the ACM in the early 1960s, and read the monthly issues of *Communications* avidly. At the time, the term "Association for Computing Machinery" was appropriate, and the articles published in CACM reflected developments in mechanical, analog, and digital computations.

In 1967, when I was doing research at the University of Grenoble, France, I had my first paper published in CACM. It was actually the cover article, which contained what is now known as execution profiles. The impact of this first article in the computing community was immediate; many of the outstanding computer experts bombarded me with questions about details of the techniques I had used. It was obvious then that CACM was already the premier publication of computer specialists.

In 1968, after a stint at MIT, I became assistant professor at Brandeis University. Undoubtedly, having several articles published in CACM and other ACM publications counted significantly toward my promotions to higher echelons in academia. Some of these articles were co-authored with my talented undergraduate students who gained invaluable experience in participating in my research; they also witnessed first-hand the efforts needed to have a paper published in a top academic journal.

I recall that during the 1970s one could hope to understand the material in CACM from cover to cover, even though the corpus of knowledge in computers was growing fast. Throughout both that decade and the next, CACM continued publishing major research articles that detailed the gems of achievements in computer science. Since then, the field has mushroomed, and specialization has become a must. The establishment of ACM Journals and Transactions dealing with specialized topics followed this trend. It was then essential to screen out the articles submitted to CACM that were more appropriate to other journals so as to achieve a balance that promoted the work done in many areas of computer science. This was no easy task and I am sure that many worthy articles did not make it to the pages of CACM.

In the early years ACM depended heavily on



THE JULY 1993 ISSUE OF *COMMUNICATIONS* WAS HONORED WITH THE BEST SINGLE ISSUE AWARD BY THE ASSOCIATION OF AMERICAN PUBLISHERS.

volunteer work to select, referee, revise, and publish papers. The momentous increase in the volume of publications eventually led ACM headquarters to recruit the help of specialists in publishing, as is done in most professional societies.

It was in this environment that Peter Denning, a recognized and experienced colleague, was appointed EIC of *Communications*, and James Maurer, a seasoned scientific publisher, was hired as Executive Editor (EE). It was Jim who, in the mid-1980s, invited me to join *Communications*' Editorial Board as one of its Associate Editors (AE).

The experience I gained during my five years as an AE was invaluable. I firmly believe this position is the best training ground for potential EICs. Since the choice of referees for a submitted paper is one of the important tasks delegated to an AE, he or she obviously plays a hand in the acceptance or rejection of the paper. For an AE, tricky choices abound; for example, in selecting the referees for a paper authored by a well-known researcher, the AE's role was more like a judge than a researcher!

Remember this was pre-Web, pre-Google. Indeed, there was not even a well-designed database in the late 1980s where an AE could check for related articles, record the names of authors and paper titles, list the referees, or note mailing dates. There was no automatic means of prompting reviewers to get their job done on time! Everything was done *manually*.

Often, after a paper is finally reviewed, an AE is confronted with conflicting advice from the referees, and a decision had to be made as rapidly as possible about acceptance, revision, or rejection. Editors, like judges, are human beings and subject to controversial decisions. My approach as an AE has always been to reply to an author of a rejected paper with constructive criticism and, whenever appropriate, suggest submission to related journals that cater to the topic at hand.

In 1992, Jim Maurer and Peter Denning invited me to become the EIC for CACM. As with any candidate for a position of high responsibility and visibility, I wondered if I would be up to the task. After reflection, I decided to accept the challenge. The acceptance was followed by a steep learning curve. Diane Crawford, the present EE, had just been appointed as the editor replacing Jim Maurer. My main goal was to join her as a partner in carrying on the task of keeping CACM as the leading ACM publication. At that time, the Association had over 80,000 members who received the monthly issues of CACM; the editors were responsible for ensuring a constant and timely stream of high-quality articles catering to readers with diverse backgrounds.

As EIC, I was also asked to join the ACM Publication Board, chaired by Peter Denning. I have seldom participated in such focused and active board meetings. Under Peter's leadership the board accomplished one of the ACM's grandest plans: the establishment of its now unreservedly success-

ful Digital Library. Great credit should be given to the board members at that critical time, including Hal Berghel, John Clippinger, Bill Gruener, Marvin Israel, Wendy MacKay, Christine Montgomery, Peter Wegner, and Gio Wiederhold, who were joined later by Bill Arms, Peter Polson, David Wise, and Ron Boisvert.

My four-year tenure as EIC of CACM can be best described as a transition period after which the EE would have global control over the material published in the CACM. This seemed to me inevitable, since the task of keeping a steady flow of articles representing the efforts of an entire community was beyond the scope of a single volunteer EIC.

Decisions and actions are always taken within certain contexts and an organization must be flexible and dynamic to cope with new environments. As such, the decision to have Moshe Vardi assume the re-created position of EIC will likely open new horizons for CACM. Moshe is a dynamic and well-recognized member of our community. We wish him success in carrying out his new responsibilities.

Finally, I want to say a few words about the experience I gained as a decision maker in nonprofit organizations like Brandeis and the ACM. I find it extremely important to understand the duality between idealism and pragmatism. One of these components cannot survive without the other. As an academician and researcher my initial inclination was toward idealism. When I assumed administrative and managerial positions, such as department chair or the editorship of CACM, I immediately recognized the need to be among pragmatists. I believe that successful organizations balance idealism with pragmatism and maintain a healthy tension between the two.

At this stage of its development, computer science risks fragmentation if we do not stress the basic concepts that bind its practitioners. At the same time we cannot ignore the complexity of the world that surrounds us and drives us toward interdisciplinary pursuits. Tomorrow's computer scientists will have to navigate wisely around the extreme of hyper-specialization while pursuing new frontiers in computer science. As we go forward CACM will continue to be the ideal venue to help our community develop new ideas and stay rooted in the basic tenets of computer science. C

JACQUES COHEN (jc@cs.brandeis.edu) is the TJX/Feldberg Professor of Computer Science at Brandeis University, Waltham, MA.

IN RECOGNITION

The former EICs of *Communications* would like to recognize the efforts of the Executive Editors who have supported us over the years. The journey began with Myrtle R. Kellington at the helm. Her high standards and devotion to putting out a good product were of critical importance to ACM's publications effort. This high standard has been continued by the outstanding EEs who followed. Our thanks to:

Myrtle R. Kellington (1958–1976)
Mark S. Mandelbaum (1977–1979),
Janet G. Benton (1980–June 1986),
James Maurer (July 1986–July 1992), and
Diane Crawford (Aug. 1992–).

BY MOSHE Y. VARDI

EIC YEARS 2008–

CACM: PAST, PRESENT, AND FUTURE

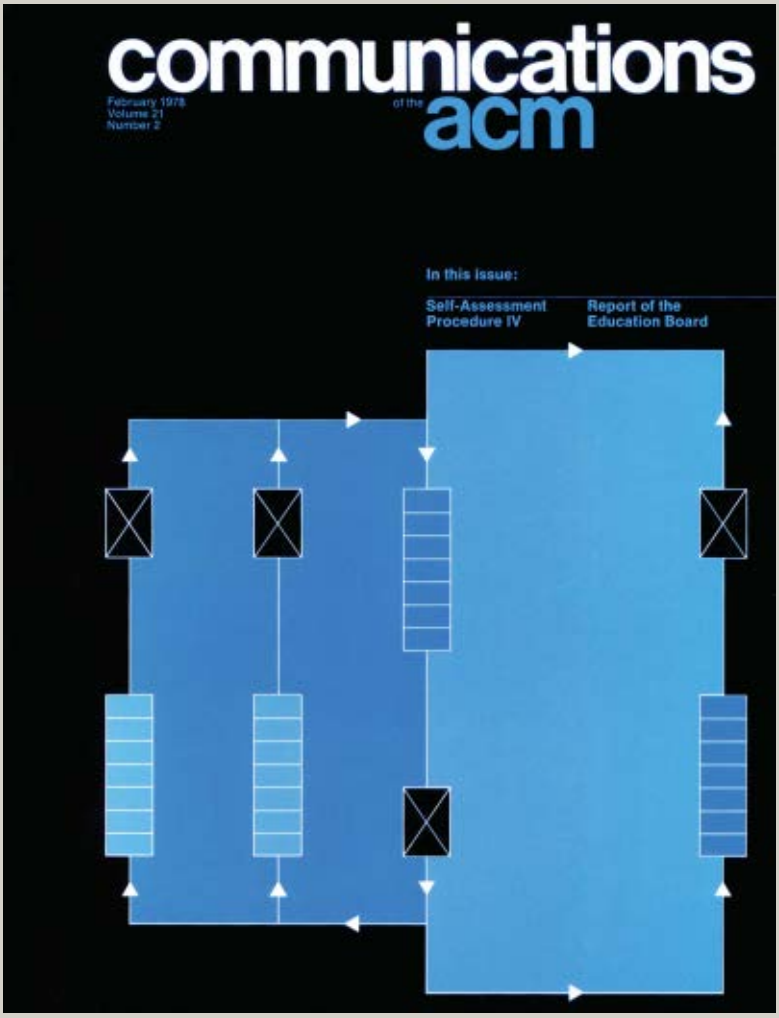
The French adage "Plus ça change, plus c'est la même chose," or, the more things change, the more they stay the same, still rings true today. Reading over the essays of my predecessors, one recognizes the thread that runs through all of them, which is the constant need of CACM to reinvent itself. In fact, I discovered an April 24, 1964 report from a Commission of Thoughtful Persons to the ACM Council that stated "It was felt that *Communications* was becoming too much of a journal and that a re-evaluation is in order." I suspect this ongoing need to rethink CACM, a flagship publication for professionals working in a fast-moving and ever-changing field, will stay with us for the foreseeable future.

I came of age as a computer scientist in the late 1970s, during my formative years as a graduate student. I remember being highly influenced by some great research articles published during CACM's "black-and-blue" years. E.F. Codd's paper, "A Relational Model of Data for Large Shared Data Banks" published in the June 1970 issue was given on Mt. Sinai, from my perspective. (1970 was felt to be in the dim past for a graduate student in 1979.) It was clear upon its February 1978 publication that Rivest, Shamir, and Adleman's paper, "A Method for Obtaining Digital Signature and Public-Key Cryptosystems," was a seminal one. And in May 1979, De Millo, Lipton, and Perlis's "Social Processes and Proofs of Theorems and Programs" was instantly controversial. (Indeed, it still makes for interesting reading today, though the tremendous progress in formal-methods research has dulled its edge.)

The essays here by Denning and Cohen describe the changes that CACM underwent during the 1980s and 1990s. In 1996, the ACM Publications Board decided that CACM ought to be largely run by professional staff, guided by an advisory board. When Cohen's tenure as EIC ended, the Board did not appoint a new EIC.

In retrospect, eliminating the position of CACM EIC and

THE FEBRUARY 1978 ISSUE OF CACM FEATURED THE LANDMARK PAPER "A METHOD FOR OBTAINING DIGITAL SIGNATURE AND PUBLIC-KEY CRYPTOSYSTEMS" BY RONALD L. RIVEST, ADI SHAMIR, AND LEONARD ADLEMAN. THE NEW CACM REVITALIZATION PLAN AIMS TO DRAW RESEARCH WORK OF GREAT INFLUENCE INTO THE FOLD.



reducing the role of the editorial advisory board left a void. By the mid-2000s, it was increasingly clear that CACM needed attention. When Dave Patterson became ACM President in 2005, "fixing" CACM became one of the priorities of his presidency. In his January 2006 "President's Letter," he wrote "When I was running for ACM President, I asked people for feedback about the Association. The consistent advice I received was to do something ... about *Communications of the ACM*." Plainly speaking, by 2005 dissatisfaction with CACM has become quite pervasive in broad segments of the computing community.

While the format envisioned in the early 1980s was that of a content-rich magazine, the reduced role of the editorial advisory board combined with a relatively small professional staff meant that most of the content came from submitted articles. Over the years those articles have evolved to be strongly slanted toward Management Information Systems. Over time, a significant segment of the ACM membership lost interest in the publication.

Patterson argued that this state of affairs was unacceptable, concluding "ACM must present a compelling value proposition for individuals to join or stay members of ACM. This means our flagship publication must be the best it can possibly be."

In mid-2005, Patterson commissioned the CACM Task Force to discuss how best to tackle the revitalization of CACM. Their conversations took into account the weaknesses within CACM coverage, the evolving composition of ACM membership, its outdated design, the emergence of the Web and the ACM Digital Library, and the introduction of *Queue*, ACM's magazine for young practitioners that debuted in 2003.

Patterson's Task Force, like the 1983 counterparts, was heavily influenced by the content model of *Science,* one the most prestigious scientific publications (being published in *Science* is a career milestone for a scientist). *Science*'s content consists of an extensive news section, several columns (policy, education, and book reviews), and research articles. While the research articles are aimed at specialists, about one-third of the articles in each issue are accompanied by a short

Re-creating the prestige of the past for CACM seems to me to be an important step in restoring the image of computing. CACM is not just the flagship publication of ACM, it is also, or at least it ought to be, the flagship publication of the computing field. CACM is our "storefront window"; it ought to project an exciting image of a dynamic field.

"Perspective" piece aimed at a general scientific audience.

The final model proposed by the CACM Task Force in late 2005 is similar to the model envisioned by the 1983 redesign effort noted in Denning's essay. It consisted of news, columns, computing practices, and research articles to be drawn from ACM conferences and journals and accompanied by brief but broadly aimed perspectives.

I was asked by Dave Patterson to take over the task of revamping CACM in early 2006. After some hesitation, due to other commitments and the magnitude of the task, I accepted the job.

The computing field went through a perfect storm in the early 2000s: the dot-com crash, the telecom crash, the offshoring scare, and a research-funding crisis. After its phase of glamour in the late 1990s, the field seems to have lost its luster. For those of us in academia, the plunging undergraduate enrollment is evidence for the "image crisis." At the same time, I fervently believe that our field has a glorious future. Re-creating the prestige of the past for CACM seems to me to be an important step in restoring the image of computing. CACM is not just the flagship publication of ACM, it is also, or at least it ought to be, the flagship publication of the computing field. CACM is our "storefront window"; it ought to project an exciting image of a dynamic field.

I love the idea of getting research articles from computing research conferences. The reliance of our field on conferences is unique among the science and engineering disciplines. The fast publication cycles and the sharp focus of the conferences move our field forward very quickly. At the same time, our conference-based culture has resulted in a severe fragmentation of computing research. A reader of *Science* can keep track of progress across all of science. In contrast, it seems impossible to keep track of progress across computing research, even at a high level of abstraction. A strong Research section in CACM might be able to help us re-create the unity that our field had in its early days, as reflected in the early CACM.

I embarked on the CACM renewal project in late 2006. With all due respect to the CACM Task Force, I was not yet ready to adopt their conclusions. The Task Force consisted of a group of ACM insiders. I thought it best to carry out a much broader conversation before committing to a particular model.

During the winter and spring of 2007, I participated in four major conversations about the "new CACM." The first conversation was with the SIG Board, which consists of the chairs of all ACM's Special Interest Groups. I then organized three focus groups—in New York, San Francisco, and London—each consisting of about 25 computing professionals from industry and academia. In each conversation I described the history of CACM and presented the recommendations of the Task Force. This was followed by very lively conversations typically lasting several hours. It turns out that people, though unhappy with CACM, do care passionately about it and do have many ideas on how to revitalize it.

There were several main points that echoed throughout these conversations. First and foremost, there was almost unanimity that ACM must have a print flagship publication. With all the advances in online publishing, there is yet no substitute for paper

publishing. The idea of bringing select research articles back into CACM was quite popular, but the participants felt strongly that such articles should be rewritten and well edited to address CACM's broad readership. At the same time, there was strong consensus that CACM should continue to publish submitted, peer-reviewed articles, although with other new content to be included in the magazine, the space available for submitted articles would be more limited and thus the articles published would need to be of the highest quality.

As far as content addressing the interests of practitioners, the general feeling was that *Queue* provides high-quality content, focusing on software practice, and CACM should not attempt to compete with *Queue*. In addition, the participants uniformly expressed a desire to see a strong news section and an "edgier" opinions section (as put by one focus group attendee: "Let us see some blood on the opinion pages of CACM"). There was a general agreement that CACM does not have the look and feel of a high-tech publication and that a graphic redesign is long overdue. Finally, there was a lively debate, and no agreement, on the issue of possibly renaming the publication. Many felt that 50-year-old name "*Communications of the ACM*" is a valuable brand; others felt a new name is needed to send a strong message that CACM is being "rebooted."

One of my main conclusions from these conversations was that the practical content offered by *Queue* is of the highest quality and likely of interest to a broad cross-section of ACM members, not just "young practitioners" for which it was originally intended. Thus, if we want CACM to serve the broad interests of the ACM membership, then it should offer the core content available in *Queue*. Thus, the Practice section of CACM should become the print outlet for *Queue*. The *Queue* editorial board should continue to produce high-quality content and develop further the *Queue* brand through an enhanced presence on the ACM Web site to serve practitioners.

With the decision to bring *Queue* articles into CACM to address the needs of practitioners, the content model for the new CACM fell into place. The focus will be on cutting-edge material, organized in five sections: News, Practice, Breakthrough Research, Refereed Articles, and Opinions and Views.

News: The news section will have a very distinct "voice," covering research and practice in computing on a global scale. There will be an increased emphasis on research from a news perspective. As a monthly publication, CACM will not compete with more timely news services, but rather analyze the latest news in greater depth for computing professionals and interpret its potential impact. ACM news of broad interests will also be covered.

Practice: CACM will offer coverage of cutting-edge, emerging technology issues. Such material would be of interest to both academics and practitioners. To obtain this material, the *Queue* editorial board, led by Stephen Bourne, will continue to function as it has in the past—providing articles, columns, and interviews.

Breakthrough Research: The goal is to bring research articles, covering the broad spectrum of computing research, back to CACM. This will be a combined effort of conference and program chairs (ACM as well as non-ACM) and the CACM editorial board to select the best research material coming out of conferences and share that information with the wide readership of CACM. Each selected research paper will be adapted to the broad CACM readership and will be preceded by a short "Technical Perspective," giving readers an overview of the important points and significance of the research. These perspectives will be written by noted experts in the field. (This issue of CACM contains two sample research articles beginning on page 104 that span the spectrum of computing research: an article on MapReduce, with perspective by David Patterson, and an article on locally sensitive hashing, with perspective by Bernard Chazelle.)

Refereed Articles: CACM will continue to publish, as it has since its early days, peer-reviewed articles. Such articles must be of the highest quality and of interest to a broad sector of the ACM membership. We will seek both solicited and unsolicited submissions. In particular, the CACM editorial board will solicit survey and review articles in topics of broad interest.

Opinions and Views: The final component in the content model of the new CACM is a section dedicated to opinions

and views, typically of a nontechnical nature. Feature columns from respected voices in computing have always been a very popular element in CACM. The plan is to maintain the most popular columns, as well as add new dynamic ones. In addition, the editorial board will continually solicit opinion pieces on topics of broad interest to the computing community. Controversial issues will not be avoided, but be dealt with fairly, by representing both sides of the issue. To maintain the freshness of this section, the editorial board will seek, on an ongoing basis, new material in areas such as global issues, ethics, history, education, policy, interviews with luminaries, and the like.

"Healthy soul in a healthy body," said the ancient Greeks. To that end, CACM will embark on a complete graphic overhaul; from cover logo to career opportunities and everything in between. All these very different elements must have a cohesive, professional, high-tech look and feel. There is also a need for a complete redesign of CACM's Web site, with the goal of creating a dynamic, rich Web site that is much more than an online store of the magazine's content. The aim is to think of CACM as consisting of a print publication, a rich Web site, and email channel to readers.

The reader who compares the content model proposed here with the content model described in Denning's essay would be struck by the similarity. A natural question is whether this model would be successful now, when it has not been so successful in the past. In my opinion, previous redesigns of CACM tended to put more attention on the content and look and feel, but less attention to developing the organizational structure that can support the production of a high-quality monthly publication. By way of comparison, CACM is produced by a staff of five professionals, while *Science* is produced by a staff of 100 professionals! By harnessing the current staffs of CACM and *Queue,* and by hiring additional professionals with critical skills, we will have the professional staff that ACM's flagship publication deserves.

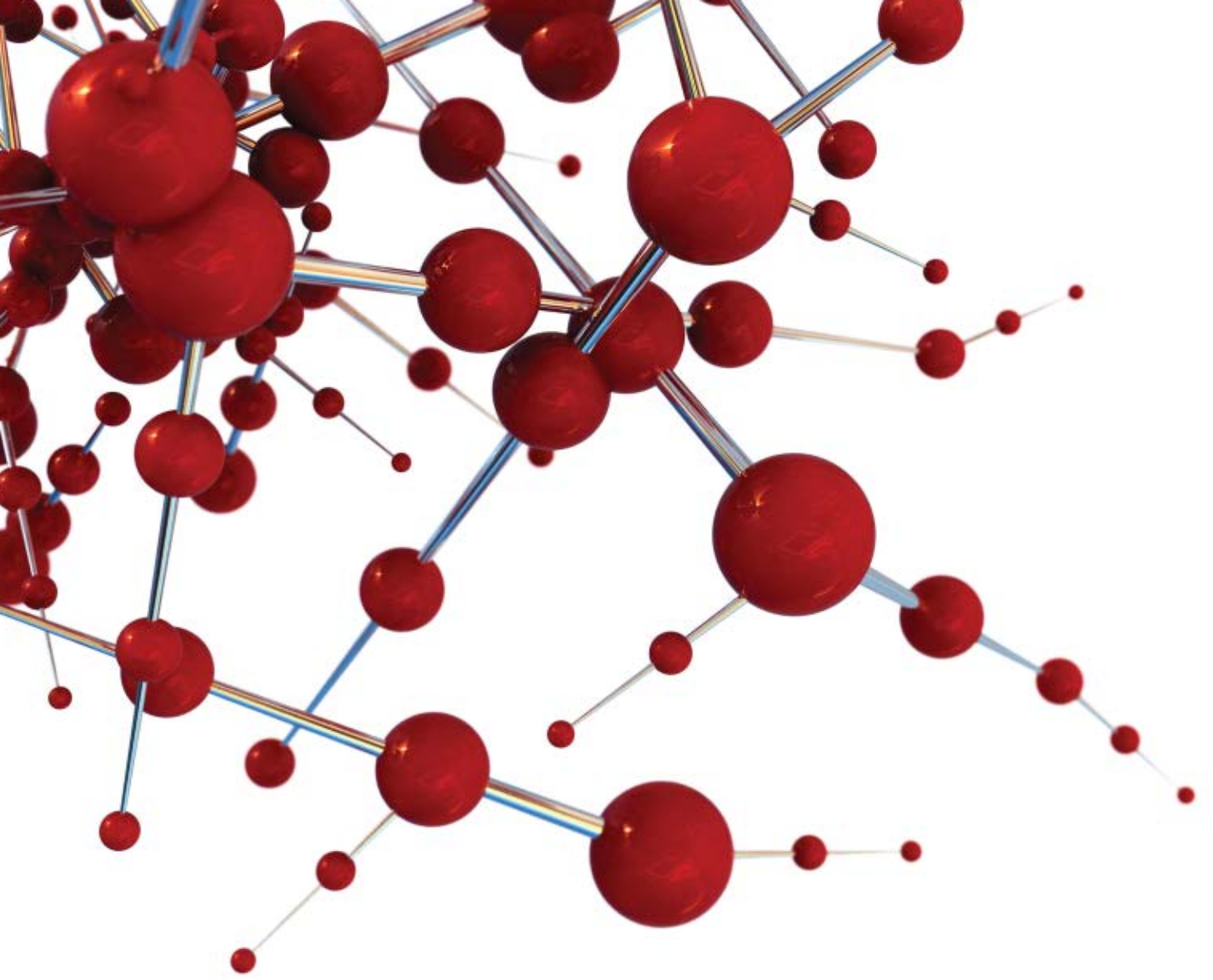
For CACM to stay attuned to the interests of the computing community, it needs an active and authoritative editorial board. One must be mindful that editorial board members are volunteers, who typically shoulder the commitments of a full-time job. While it is tempting to conceive of a small cohesive board, the burden of producing a monthly publication would overwhelm a small volunteer board. The solution is to constitute the board as a collection of semi-autonomous sub-boards, corresponding to the various sections of the publication. The full editorial board is expected to have around 50 members.

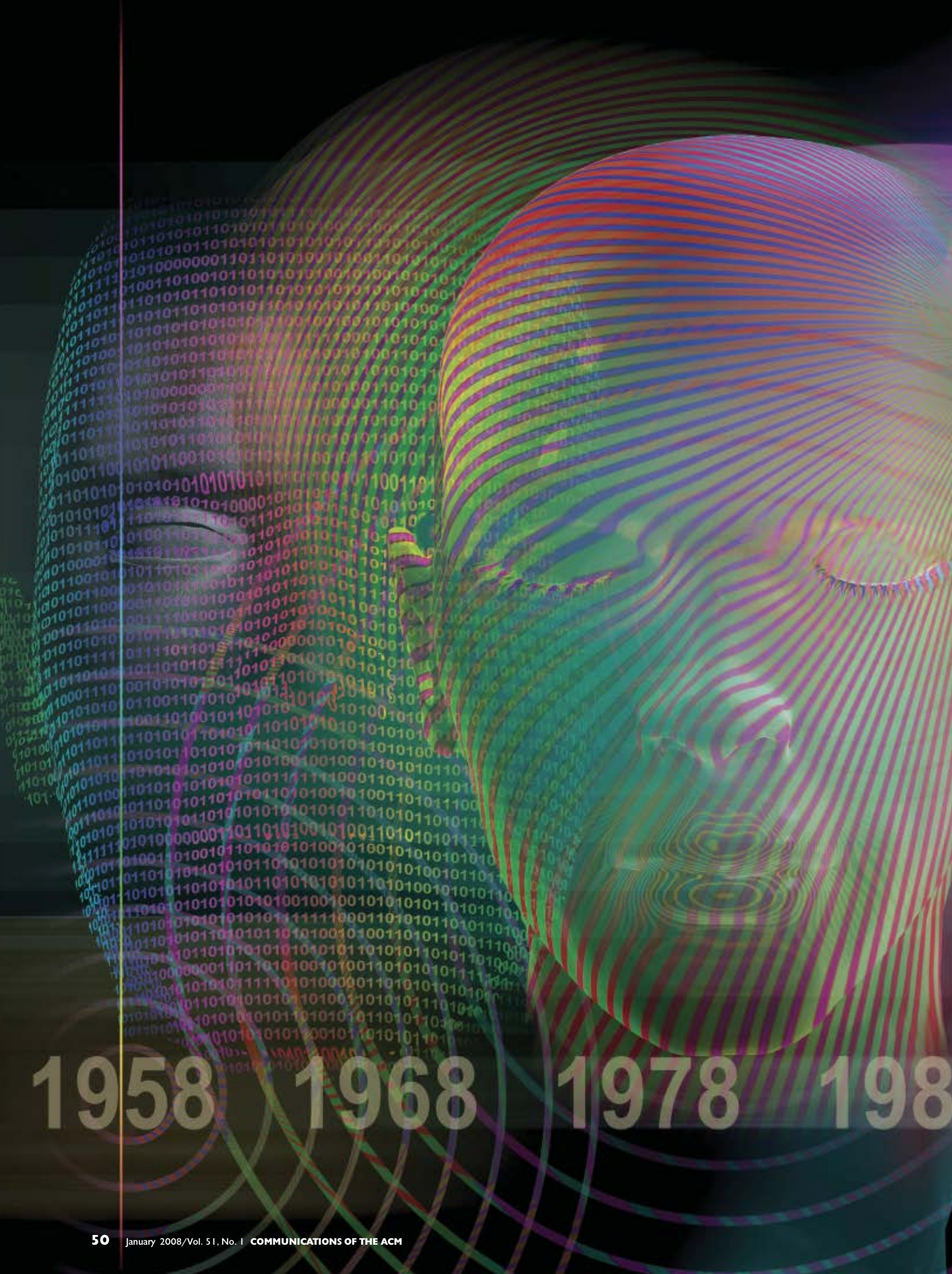
This plan for a revitalized CACM was presented to and approved by ACM Publications Board in May 2007 and ACM Council in June 2007. Staffing and recruiting for the editorial board are currently under way. We have recruited an impressive panel of sub-board chairs to date: Breakthrough Research—David Patterson and Stuart Russell; Refereed Articles—Al Aho and George Gottlob; Opinions and Views—William Aspray and Nigel Shadbolt; News—Marc Najork and Prabhakar Raghavan; and for the CACM Web site—Marti Hearst and James Landay. The launch of the redesigned CACM is planned for later this year. Watch for an announcement soon!

In conclusion, I'd like to add one final point. We live in a consumer society, so it is easy to evaluate products from a consumer perspective: "Is CACM a satisfactory product?" "Am I getting my money's worth for my ACM membership?" ACM, however, is not a consumer-product vendor, it is a professional society. We are not ACM customers, we are ACM members. CACM is not a product, it is a project. For this project to succeed, the membership of ACM must collectively undertake it. Let us—together—make CACM the exciting publication it should be. Please write to me at cacm-eic@acm.org. C

Moshe Y. Vardi (cacm-eic@acm.org) is Karen Ostrum George Professor in Computational Engineering and Director of the Computer and Information Technology Institute at Rice University, Houston, TX.

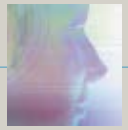
The Research Preview begins on page 104




1958
1968
1978
198

VOICES

Many of the leading voices in computing have graced the pages of *Communications* over the past 50 years. Their research, opinions, and professional journeys have influenced countless future leaders, and will continue to do so. In this section, we've invited some of the most prominent among them to look into the future of the field and reflect on how 50 years of CACM are helping inspire and create it. As difficult as it is to look even a few years ahead, these voices have managed to anticipate the needs of future users, even as they advance the field's deepest scientific principles. A notable theme is how much computing ultimately depends on trust, whether in an e-commerce transaction, a robot's behavior, a software agent's instructions, a network's architecture, a human's intentions, or the scholarship of a paper. See how computation connects us all.



BY JON BENTLEY

IN THE REALM OF INSIGHT AND CREATIVITY

The intellectual pleasures and financial rewards of solving one programming problem, it turns out, are just the prelude to solving many more.

The first copy of *Communications* to arrive in my mailbox was the glorious July 1972 issue celebrating 25 years of the ACM in a 200-page overview of computing. I was a sophomore at Long Beach City College, and it introduced me to the academic field of computing, as well as to three luminaries for whom I would find myself working over the next decade: Donald Knuth, Joe Traub, and Sandy Fraser. From my very first experience, the magazine was performing its critical role: communicating to members of the ACM the ideas that would challenge and change their professional lives.

I eagerly awaited the arrival of each subsequent issue and still vividly recall many articles from those years. Edsger Dijkstra encouraged me to respect the difficulty of our craft in "The Humble Programmer" (October 1972). David Lorge Parnas taught me fundamental principles in "On the Criteria to Be Used in Decomposing Systems into Modules" (December 1972). And Dennis Ritchie and Ken Thompson introduced me to "The Unix Time-Sharing System" (July 1974) that would soon be my home. The list goes on: Donald Knuth on "Computer Programming As an Art" (December 1974), Brian Kernighan and Lorinda Cherry on "A System for Typesetting Mathematics" (March 1975), and Robert Floyd and Ronald Rivest on "Expected Time Bounds for Selection" (March 1975). In those pages I first met the ideas that would mold my professional career and the people who would become my colleagues, mentors, and heroes.

I cannot describe the excitement I felt when my own first article "Multidimensional Binary Search Trees Used for Associative Searching" was published in September 1975. It received second place in the 1975 Forsythe Student Paper Competition. Back then, original research papers were viewed as within the reach of undergraduate students (truth be told, I tied with a high school student) and of interest to the general ACM community. I wrote my paper (under

the guidance of Donald Knuth) explicitly for the Competition; had it not existed, I would not have written it. It remains to this day my most cited research paper. (And because I'm sure that your inquiring mind wants to know, I'll point out that first place went to a lad by the name of Guy Steele, who I'm betting turned out okay.[1]

Over the next few years, I published several technical papers in CACM. We submitted only papers we felt were of broad interest to the computing community and that had a high ratio of ideas to pages. I

[1]He's been a fellow of Sun Microsystems since 2003 and works on Sun's Programming Language Research project.

In those pages I first met the ideas that would mold my professional career and the people who would become my colleagues, mentors, and heroes.

remain particularly proud of "Multidimensional Divide-and-Conquer" (April 1980), summarizing my Ph.D. thesis, and "A Locally Adaptive Data Compression Scheme" (April 1986) with Daniel Sleator, Robert Tarjan, and Victor Wei, introducing a method for compressing text files.

Programming has been very, very good to me. I wrote my first Fortran program as a junior in high school in 1969 and have now followed my bliss in this field for almost four decades. Programming has provided a livelihood, intellectual challenge and reward, deep friendships, and numerous joys indescribable. Of all the good things that have happened in my career as a programmer, though, one of the best was and is writing the "Programming Pearls" column in CACM beginning August 1983. These columns presented neither new research nor systematic reviews of the field. I would later describe them as "programming pearls whose origins lie beyond solid engineering, in the realm of insight and creativity." For the most part, they were fun stories of how clever colleagues had phrased and solved programming problems, not infrequently discovering that "we have met the enemy, and he is us." When Peter Denning graciously offered me the opportunity to write the column, he pointed out that each of my scribblings would land on "the 70,000 most important coffee tables in the world of programming."

The best columns were the ones in which master programmers shared helpful insights with their colleagues. In March 1984, Bob Martin's "The Back of the Envelope" showed how even small estimates are useful in designing big computer systems. The September 1985 column "Bumper-Sticker Computer Science" collected aphorisms sent in by readers; among them are these timely gems:

- The sooner you start to code, the longer the program will take (Roy Carlson);
- Allocate four digits for the year part of a date: a new millennium is approaching (David Martin); and
- Pi seconds is a nanocentury (Tom Duff).

I enhanced and collected 13 of the columns into the book *Programming Pearls* in 1986 and a second edition in 2000. We published a second collection called *More Programming Pearls: Confessions of a Coder* in 1988. The books have sometimes been used in schools, but their most important readership has always been working programmers who want to learn more about their craft.

I have long been resigned to the fact that whether I like it or not (fortunately, I usually do), I will almost always be introduced in computing circles as "Jon Bentley, who wrote 'Programming Pearls' in *Communications* in the 1980s." I remain deeply grateful to Peter Denning and Stuart Lynn who first proposed the column, to my colleagues and managers at Bell Labs who supported the work, to the programmers who contributed so much of the column's best content, and to the readers of CACM who frequently took the time to say thanks.

When I first read CACM in 1972, I was young, and it was old. Now the tables have turned: I am an old computer programmer, and it remains fresh and vital. It has done this by sticking to its primary mission: communicating appropriate content to the members of the ACM. If it stays this course, I am confident that its second half century will be as much fun and as fruitful as its first. C

Jon Bentley (jbentley@avaya.com) is a research scientist in the Software Technology Research Department of Avaya Labs Research, Basking Ridge, NJ.

BY WHITFIELD DIFFIE

INFORMATION SECURITY: 50 YEARS BEHIND, 50 YEARS AHEAD

Trust among people and organizations will be even more critical in securing communications and commerce in the future networked environment.

What was the state of information security—the combination of computer and communication security—as *Communications* first went to press? Cryptography was both secret and primitive, able to protect the confidentiality of communications but unable to perform most of the functions we ask of it today. Computer security was nonexistent.

Information security today is a vast field, with more money, publications, and practitioners than all of computer science had a half-century ago. Cryptography is largely public and becoming a standardized part of the infrastructure. Computer security is not so settled but has made great strides since its birth in the 1960s and is an important aspect of day-to-day computer operations.

Where is information security going? Away. Today it would be possible to say that you did a computation securely if you did it entirely on your own computers and if you protected them appropriately.

But we live at the end of the era of isolated computation. Within the next decade Web services will have created a vast infrastructure of companies and other organizations that can perform your most important computations faster and cheaper than you could ever do them for yourself, just as Google can search better than you can. You'll be unable to stay in business without using them but also unable to conceal your secrets from them. All the cryptography, bounds checking, and packet filtering will still be there, but the main mechanism of information security will be contractual.

How did we get to this situation? In 1958 computer security would have been very difficult to distinguish from the security of the computer itself. Computer rooms were guarded, operators and users

were vetted, and card decks and printouts were locked in safes—all physical or administrative measures. Process confinement, kernelized operating systems, and formally specified programs were all a decade in the future.

Cryptography, in the limited quarters in which it was known and practiced, would have been more recognizable but very primitive. From World War I, when mechanized cryptography got its real start, through World War II, most military cryptography was mechanical, performed by electromechanical machines whose action combined a number of table lookups with modular arithmetic. A character to be enciphered was put through a sequence of table lookups interspersed with the addition of keying characters—a slow process capable of encrypting teletype traffic but utterly inadequate for coping with voice.

The 1950s were dominated by the effort to bring the speed of encryption closer to the speed of the modern world. Military cryptography—to the degree it had gone beyond rotor machines—consisted primarily of what are called long-cycle systems. The backbones of these systems were linear feedback shift registers of maximal period. Two techniques were used to make the output (which was to be XORed with the plain text) nonlinear. The registers stuttered (paused or skipped states about half the time), and the output came from tapping a number of stages and combining them into one bit with nonlinear combinational logic.

Only one small laboratory, the Air Force Cambridge Research Center (military but out of the mainstream of cryptography), had begun looking at the ancestors of the U.S. Data Encryption Standard and many other systems while working on cryptographic techniques for identification friend or foe, the technique by which a fire-control radar recognizes that an incoming plane is friendly and should not be fired on. The radar sends the plane a challenge; if the plane decrypts the challenge, modifies it in an agreed-upon way, and reencrypts it correctly, the radar tells the gun to hold its fire.

The process of recognizing a signal by its correct encryption is one to which the stream ciphers of communications are ill suited. Rather than a system in which each bit of the message depends on one bit of key, with which it was XORed, a system is needed in which every bit of output depends on every bit of input. Today we call such a system block ciphers or electronic code books.

Over the past 50 years, both computer security and cryptography have made great strides, and CACM has played an important role in the growth of each. Computer security as we think of it today was the offspring of time sharing and multiprocessing. Once a computer could run jobs on behalf of several users at a time, guarding the computer room was no longer sufficient. It was necessary to guarantee that each individual process inside the computer was not spying on another such process.

Time sharing was born in the early 1960s and by the late 1960s was a major force. It was in use in computing laboratories around the world and offered commercially by service bureaus that, five years earlier, had been running one program at a time for their customers who submitted decks of cards. The turn from the 1960s to the 1970s marked the birth of both computer security and the modern era in cryptography.

Computer security came first. The introduction of timesharing had been particularly disruptive in the culture of military laboratories. Time sharing allowed those doing unclassified work to move into a crude approximation of the environment we enjoy today—15-character-per-second model-35 teletypes, then primitive cathode-ray tube screens rather than high-speed flat-screen displays—but interactive work within one's own office during normal working hours. Those dependent on classified computing found themselves ghettoized into working in the computer area for a few hours in the evening after the others had gone home. The result was a major program to produce far more secure computers. The formula was simple, starting with writing better code. As we envisioned it then, this meant mathematically proven to be correct. But as not all of one's code can be one's best code, less-trusted code had to be confined so it couldn't do any damage. These are problems on which much time has been expended yet still have no fully satisfactory solution.

Curiously, computer security in the late 20th century was rescued by another great development of computer science—networking—particularly client-server computing. Networking brought forth the need for cryptography, a subject kept secret from and neglected by the computer science community at the time.

The 1970s saw the development of public-key cryptography, a new approach to secure communication that surmounted a long-accepted obstacle to the broad use of cryptography, that is, to communicate securely you must share a secret at the outset. Public-key cryptography made a major improvement in key management—by eliminating most of the need to transport secret keys—and made possible digital signatures. Together they improved key management, and digital signatures fostered the growth of Internet commerce in the 1990s. The appearance of public-key also sparked an explosion of public, business, and

government interest in more conventional forms of cryptography, catapulting cryptography to the position of best understood and most satisfactory part of information security.

Today, public-key cryptography has given birth to a second generation of systems, replacing the modular arithmetic of the first generation with arithmetic on elliptic curves. The U.S. Data Encryption Standard of the 1970s, an algorithm of moderate strength, has been replaced with the Advanced Encryption Standard, which may be the most secure and carefully studied algorithm in the world. Technical developments have been accompanied by a change of heart at the National Security Agency, which has embraced the public developments by adopting a "Suite B" of public algorithms as satisfactory for all levels of classified traffic. Cryptography is now the soundest aspect of information security, and modern security systems are rarely penetrated by confronting the cryptography directly.

Computer security has not fared as well. Progress developing high-assurance systems, though substantial, has not been as great as expected or required. Implementation of features in most commercial computing has taken precedence over security, and the state of Internet security is widely lamented. Real progress awaits major improvements in both construction and evaluation of computer programs.

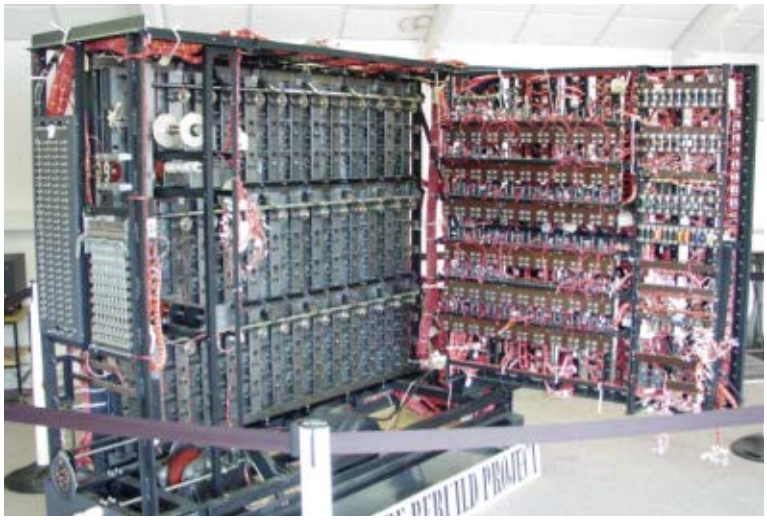
In contrast to cryptography's solid technical status and the fact that the Secure Sockets Layer is the most widely deployed cryptographic mechanism of all time, SSL effectiveness is limited. The weak computer-security foundation on which cryptography must be implemented has made it problematic to scale the key management system to Internet size.

Coupled with this is a serious human-factors failure of all security systems. The Internet is a medium in which users want to talk to both people and machines they trust, as well as to those they don't trust. Unfortunately, the means to recognize those who are trustworthy (and, say, accept upgrades to programs from them) is not available. As a result, cryptography has failed to protect us from a network in which a quarter of the computers have been captured by bot networks and half the email is spam.

What will happen over the next half century? Two great challenges loom:

True computer, communications, and network security are seen by police and intelligence agencies as an obstacle to the prevention of terrorism. Although attempts to block the use of good cryptography subsided at the end of the 1990s, a program of building-in ubiquitous wiretapping is being carried out at a pace that does not inspire confidence that the interception facilities will be secure against capture and misused by parties unknown.

More fundamental is the growth of Web services. Today, even the most security-conscious companies cannot avoid trusting their engineering and marketing directions to Google and its trade-secret techniques. The query stream reveals all of our interests, and only Google's good practices and reputation guarantee they are not put at the service of competitors. Much sooner than the next half century, Web services will have destroyed locality in computing.

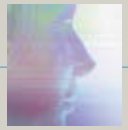


BRITISH CRYPTOLOGISTS USED AN ELECTROMECHANICAL DEVICE CALLED THE BOMBE, DESIGNED BY ALAN TURING AND REBUILT HERE FOR THE BLETCHLEY PARK MUSEUM, TO HELP BREAK CODE SIGNALS FROM THE GERMAN ENIGMA MACHINE DURING WORLD WAR II.

No significant corporate computation will take place on any one organization's machines. Programs will look at various yellow pages and advertisements and choose the most cost-effective providers for their most intensive computations. Image rendering, heat flow, marketing campaign modeling, and a host of services not yet imagined will be provided by myriad companies offering proprietary solutions.

When this happens, what we call secure computation today—you did it on your own computer and protected it adequately—may be gone forever. C

Whitfield Diffie (whitfield.diffie@sun.com) is the chief security officer, a vice president, and a Sun Fellow of Sun Microsystems, Mountain, View, CA.



BY JEANNETTE M. WING

FIVE DEEP QUESTIONS IN COMPUTING

Even if they seem unanswerable, just trying to answer them will advance the field's scientific foundations and help engineer the systems we can only imagine.

The field of computing is driven by boundless technological innovation and societal expectations. The field runs at such a maddening pace that we barely have time to pause and enjoy the ride. The time between an ingenious idea springing from a research laboratory or coffeehouse conversation and its emergence as a product or service is so short and the frequency of the commercialization cycle of ideas so great that we rarely stop to savor even our own successes.

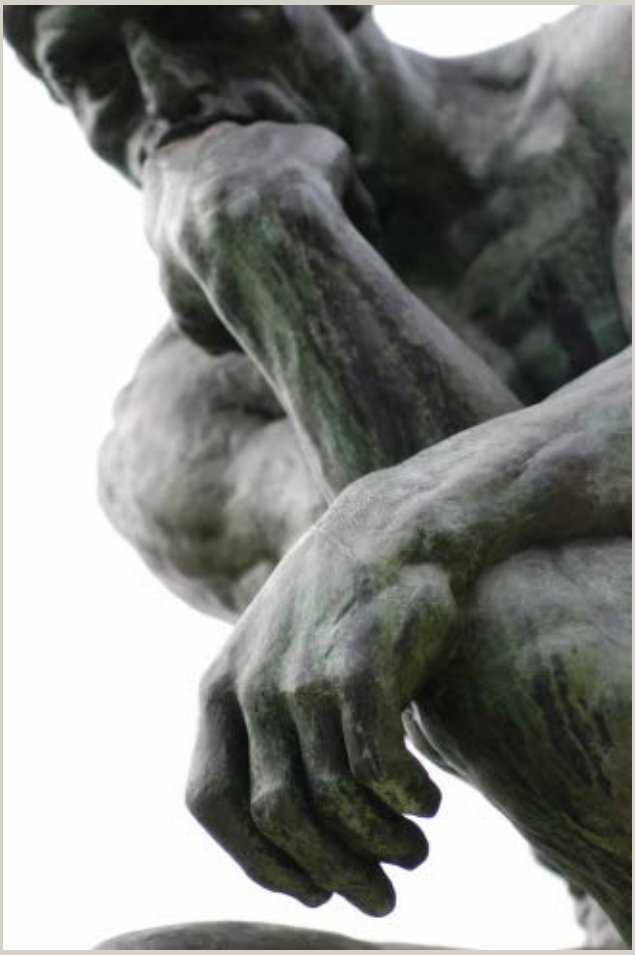
While it is easy to be swept away by the cool things we do, we should not forget that the field also contributes to fundamental scientific knowledge. So let's take a step back from the frenzy and think about the science computing pursues. To help, I pose five deep questions [3]:

P = NP?
What is computable?
What is intelligence?
What is information?[1]
(How) can we build complex systems simply?

There is nothing special about the number five; it is just a way to get a list going. I call them deep because they speak to the foundations of the field, reflecting the kind of far-reaching issues that drive day-to-day research and researchers toward understanding and expanding the frontiers of computing.

The question of whether P equals NP is undeniably the most well-known unsolved problem in the field. A proof in the positive (P = NP) would have profound practical consequences, shaking the founda-

[1]I thank Robert L. Constable, Dean of Faculty of Computing and Information Science at Cornell University, Ithaca, NY, for suggesting this question.



tions of cryptography upon which today's electronic transactions are based. It could give us efficient ways to solve intractable problems (such as the traveling salesman problem and the subgraph isomorphism problem) that arise in mathematics, as well as in every other science and engineering discipline. A proof in the negative ($P \neq NP$) would have profound theoretical consequences for computer science and mathematics, perhaps by discovering a brand-new proof technique.

In order to answer what is computable, we must consider the underlying machine (abstract or physical) that is the computer. Consider the Internet as a computer. What is a universal machine model for the

Is there a complexity theory for analyzing our real-world computing systems as there is for the algorithms we invent?

Internet? Consider a molecular computer, a DNA computer, a nano-scale computer, or even a quantum computer [1]. What problems can and cannot be solved through them? If contemplating these emerging substrates is not mind-bending enough, consider a human and a machine working together as a single computer to solve problems that neither would be able to solve alone [2]. Given that humans and machines have complementary computing capability, now ask: What is computable?

In the 1950s, the founders of artificial intelligence challenged computing researchers with the third question. As our understanding of human speech, vision, language, and motor skills has improved since then, and as we have achieved scientific and technological advances in computer science, neuroscience, cognitive science, and the behavioral sciences, the landscape has changed dramatically. Computer scientists can now probe both deeply and broadly in our quest to understand intelligence, from the neuron to the brain, from a person to a population.

"Information" in the field of computing has seemingly disparate meanings depending on scientific context: information theory, information processing, information retrieval, or information science. Distinguishing signal from noise is relevant in all these contexts, whether we mean transmitting bits over a wire, searching for an answer through the Web, or extracting knowledge from an ocean of sensor data. In essence, there is a chain of representations, from bits to data to information to knowledge. Beyond computing, nature has its own way of encoding information that is not as simplistic as using 0s and 1s. The genetic code is an obvious example. More sweepingly, by interpreting a DNA strand, a cell, or an organism as a reactive system (processing inputs from its environment and producing outputs that affect that environment), it is no longer metaphorical to say biology is an information science. Ditto geosciences. Meanwhile, with quantum computing, it's not bits but qubits.

Our engineering prowess creates computer, communication, and information systems that enhance everyone's daily lives and enable us to do astonishing things: instantaneous access to and sharing of information through palm-size devices with friends in social networks of tens of millions of users; dance with remote partners through 3D tele-immersion [4]; and lead alternative lives through avatars that can even defy the laws of physics in virtual worlds. The complexity of these systems delivers the richness of functionality we enjoy today, with time and space performance that spoil us. Their complexity, however, also makes it difficult for even the original system developers to analyze, model, or predict system behavior, let alone anticipate the emergent behavior of multiple interacting systems.

Can we build systems with simple and elegant designs that are easy to understand, modify, and evolve yet still provide the functionality we might take for granted today and dream of for tomorrow? Is there a complexity theory for analyzing our real-world computing systems as there is for the algorithms we invent? Such a theory would need to consider measures of not just time and space but of energy and cost, as well as dependability, security, and usability, most of which elude quantification today. More ambitiously (or crazily), is there a complexity theory that spans both the theory and practice of computing?

I pose these questions to stimulate deep thinking and further discussion. What deep questions about computing would you want answered? In 50 years, how different will they and their answers be from what we ask and are able to answer today? C

References

1. Reiffel, E. and Polak, W. An introduction to quantum computing for non-physicists. *ACM Computing Surveys 32,* 3 (Sept. 2000), 300–335.
2. von Ahn, L. Games with a purpose. *IEEE Computer* (June 2006), 96–98.
3. Wing, J. Thinking about computing. *Computing Research News 19,* 5 (Nov. 2007).
4. Yang, Z., Yu, B., Wu, W., Diankov, R., and Bajscy, R. Collaborative dancing in tele-immersive environment. In *Proceedings of ACM Multimedia* (Santa Barbara, CA, Oct. 23–27). ACM Press, New York, 2006, 723–726.

Jeannette M. Wing (jwing@nsf.gov) is Assistant Director of the Computer and Information Science and Engineering Directorate at the National Science Foundation, Arlington, VA, and the President's Professor of Computer Science in the Computer Science Department at Carnegie Mellon University, Pittsburgh, PA.

BY EUGENE H. SPAFFORD

INSPIRATION AND TRUST

Not every important problem can be solved through science and technology, but that doesn't mean they shouldn't be addressed.

In August 1984, *Communications* published one of the most important works in the literature on information security and assurance—the Turing Award essay "Reflections on Trusting Trust" by Ken Thompson [3]. In a concise and elegant manner, Thompson presented what may be the fundamental reason why real-world cyber security is so difficult: At some level we must trust that what we are using is correct because we cannot verify it. Furthermore, the essay embodied other important points, including the problem of the insider threat, as well as the lesson that technology alone cannot address all the problems of security. It is no wonder that it is on every significant "required reading" list concerning security, privacy, and assurance and has served to inspire so many professionals to get as close as they can to solutions to these problems.

I am one of those people whose career was changed by the opportune appearance of that particular essay. In 1984 I was developing the second generation of the CLOUDS distributed kernel for my Ph.D. thesis at Georgia Tech, where I was also helping administer some of the machines that were part of the early NSFnet and Usenet. Those experiences impressed on me the difficulty of configuring systems correctly to protect against intruders and exploitation. When queried, my faculty advisors steered me toward the extant literature in security, which dealt largely with cryptography, covert channels, and capability architectures. These topics didn't give me much insight into how to protect our current operational systems, nor did they seem to suggest that such lines of inquiry might be of longer-term academic interest.

One advisor told me I was wasting my time "playing" with security. The emergence of computer viruses and major intrusions, such as the one detailed in the "Cuckoo's Egg" incident (my server was among the victimized and is why I am in the references here [2]), gave me firsthand experience with these emerging threats. It was clear to me that security issues were important, even if some of my professors didn't share that view.

This was the context in which the August 1984 issue appeared. Not only did Thompson's essay address some of the same questions I found interesting and vex-

ing but in only a few pages reflected a level of complexity I had yet to consider. Best of all, it was by someone who was firmly involved in writing real operating systems (Thompson was a co-inventor of Unix), as well as being a response to a Turing Award. This short essay validated my interest in "playing" with security and influenced my career in the years to come.

When I first joined ACM 30 years ago as an undergraduate student at the urging of another of my professors, I wasn't sure what to expect from my membership. I soon discovered my subscription to CACM to be one of its greatest benefits. In part, the world of inquiry reflected in its articles and columns (and in some of the SIG newsletters) reinforced my decision to attend graduate school. CACM revealed problems and issues that never came up in my classes but that I recognized as worthy of greater thought. I wanted to be involved in addressing some of them.

The "positions available" section in each issue also encouraged me in my annual quest for student loans; the prospect of a productive career that would (eventually) pay off those loans was reassuring.

While in graduate school at Georgia Tech working on a Master's and then Ph.D. degree in operating systems, I continued to be interested in what was going on across the discipline, and CACM provided great exposure to the challenging landscape of computing. I would often take an issue with me when I knew I would have time somewhere (at, for example, the dentist's office), as it provided more interesting reading than could be found in what was normally left in the rack. One memorable occasion is when I was chastised by an otherwise entrancing inamorata because I evidenced (at the moment) more interest in those articles than in her arrival after class; I kept all my CACM issues long after we parted company, so perhaps it was indeed a harbinger, although we didn't realize it at the time.

After graduating with my Ph.D. and completing a short post-doc in software engineering, I was hired at Purdue in 1987. I kept up my background activities in applied security, along with my deep interest in the assurance problem. Thus, in late 1988 when the Internet Worm appeared, I was prepared to investigate and write about it, although I was not formally performing research in the area at the time. Furthermore, it led to my first publication in CACM [1], something I had set as a goal during my undergraduate days when I first became an ACM member.

In the years since then, the Thompson essay has continued to indirectly inspire portions of my work. My design of the Tripwire system (www.tripwire.com) in 1992, my development of the technology underlying the recent offerings by Signacert (www.signacert.com), and my research, including with my students on execution verification and forensics (spaf.cerias.purdue.edu/students.html), all relate back to the fundamental ideas in Thompson's essay. It also influenced some of my work on the Computing Curricula 91 task force [4] and other efforts in education and computing policy. I continue to believe that everyone working in computing should be familiar with Thompson's essay, as well as why he won the Turing Award.

CACM as certainly helped shape the thinking and careers of many in the field over the past 50 years, myself included. Congratulations on turning 50, and on the many lives yet to be influenced. C

References

1. Spafford, E. Crisis and aftermath. *Commun. ACM 32,* 6 (June 1989), 678–687.
2. Stoll, C. *The Cuckoo's Egg.* Doubleday, New York, 1989.
3. Thompson, K. Reflections on trusting trust. *Commun. ACM 27,* 8 (Aug. 1984), 761–763.
4. Tucker, A. Computing curricula 1991. *Commun. ACM 34,* 6 (June 1991), 68–84.

Eugene H. Spafford (spaf.cerias.purdue.edu) is Executive Director of the Center for Education and Research in Information Assurance and Security and a professor in the Department of Computer Science in Purdue University, West Lafayette, IN. He is also chair of ACM's U.S. Public Policy Committee.

BY RODNEY BROOKS

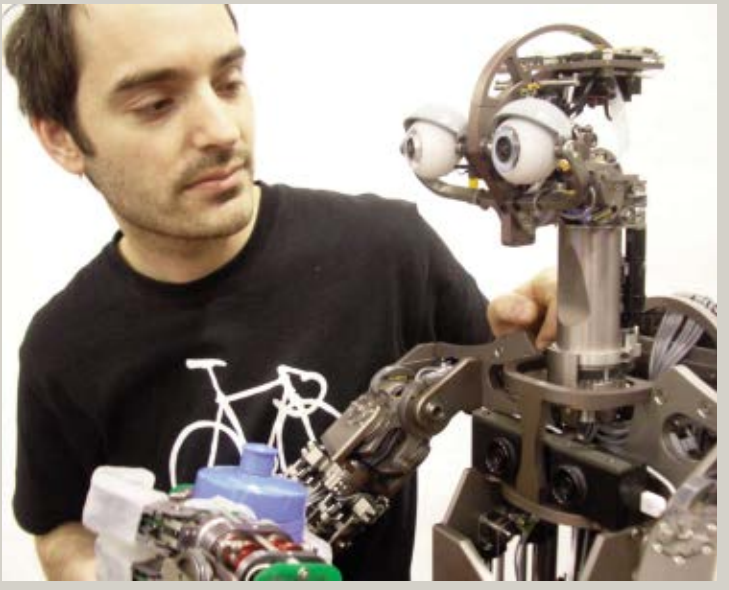
THE NEXT 50 YEARS

Expect new ways to understand computation, computational abstractions for our computing machinery, and connections between people and their information sources, as well as each other.

Since this issue celebrates 50 years of *Communications* it seems appropriate to speculate what the next 50 years may bring, in time for the 100th anniversary. For this look into the future, I cover three areas of "computing machinery": the theoretical understanding of computation; the technological substrate of computing machines; and the applications that computing machines bring to our lives.

Our current understanding of computation sprang from Alan Turing's famous 1936 paper on computable numbers but really got going around the time CACM was first published. It has centered around what is in principle computable and how much time and how much memory are needed to compute it.

By the early 1960s the idea of asymptotic analysis of algorithms was in place; if you can characterize some computation problem by a positive integer n (such as how many records you want to sort on some key) then what is the theoretical minimum time and/or storage space as a function of n required to complete the task? Even better, what is the algorithm for doing it? (In the case of sorting records, the time requirement is proportional to n*(log n), and the space is proportional to n.) This approach to understanding computation has been wonderfully productive and is still yielding both theoretical insights and practical results (such as in the field of cryptography). Other



AARON EDSINGER AND HIS DOMO UPPER-TORSO HUMANOID ROBOT (29 DEGREES OF FREEDOM) DEVELOPED AT THE MIT COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE LABORATORY (WWW.AI.MIT.EDU/PROJECTS/HUMANOID-ROBOTICS-GROUP/).

The goal would be nonbrittle software modules that plug together and just work, in the remarkable way our own flesh repairs itself when insulted.

formalisms have been developed to understand distributed computing and asynchronous computation, but none has become as established or been able to yield such universal results as asymptotic complexity analysis.

That lack of progress is sure to change over the next 50 years. New formalisms will let us analyze complex distributed systems, producing new theoretical insights that lead to practical real-world payoffs. Exactly what the basis for these formalisms will be is, of course, impossible to guess. My own bet is on resilience and adaptability. I expect we will gain insights from these two properties, both almost universal in biological systems. For example, suppose we start with the question of how to specify a computation and how quickly this particular computation is likely to diverge if there is a one-bit error in the specification. Or how quickly it will diverge if there is a one-bit error in the data on which it is acting. This potential divergence leads to all sorts of questions about resilience, then to questions about program encoding and adaptability of software to hostile environments. The goal would be nonbrittle software modules that plug together and just work, in the remarkable way our own flesh repairs itself when insulted and how our bodies adapt to transplantation of a piece of someone else's liver. The dream of reliable software may follow from such a theoretical reconsideration of the nature of computation.

As for the computing machines themselves it is worth noting that all technology generally seems to have a "use by" date. Bronze gave way to iron, horses gave way to automobiles, and more recently analog television signals finally and belatedly gave way to digital television, long after digital techniques were emulating analog, following years and years of back compatibility. We're just reaching that stage with regard to the classical von Neumann architecture for a single digital computational processor. We have spent the last few decades maintaining the appearance of a von Neumann machine with uniformly addressable memory and a single instruction stream, even though, in the interest of speed, we have had multiple levels of memories (and caches to hide them) and many parallel execution units (and pipeline stalls to hide them).

As the die size of our chips is getting so small that we cannot make it smaller and maintain the digital abstraction of what goes on underneath, we have begun to see the emergence of multi-core chips. And in traditional computing machinery style, we immediately also see an exponential increase in the number of cores on each chip. Each of these cores is itself a traditional von Neumann abstraction. The latest debate is whether to make that whole group of cores appear as a single von Neumann abstraction or bite the bullet and move beyond von Neumann.

Thus we are currently witnessing the appearance of fractures in the facade of the von Neumann abstraction. Over the next 50 years we will pass the "use by" date of this technology and adopt new computational abstractions for our computing machinery.

The most surprising thing about computation over the past 50 years has been the radical new applications that have developed, usually unpredicted, changing the way we work and live, from spreadsheets to email to the Web to search engines to social-interaction sites to the convergence of telephones, cameras, and email in a single device. Rather than make wildly speculative—and most likely wrong—predictions about applications, I will point out where a number of trends are already converging.

A key driver for applications is communication and social interaction. As a result, wireless networks are increasingly pervasive and indispensable. A key medical development over the past decade has been the implanting of chips that directly communicate with people's nervous systems; for example, more than 50,000 people worldwide now hear through the miracle of a cochlear implant embedded inside their heads (running C code, no less). In clinical trials blind patients are starting to see, just a little, with embedded chips, and quadriplegics have begun to control their environments by thinking what they want to happen and having signals generated in their brains detected and communicated by embedded chips.

Over the next 50 years we will bring computing machinery inside our bodies and connect ourselves to information sources and each other through these devices. A brave new world indeed. C

Rodney Brooks (brooks@csail.mit.edu) is the Panasonic Professor of Robotics in the MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA.

BY PAMELA SAMUELSON

HACKING INTELLECTUAL PROPERTY LAW

Considering how intellectual property law has evolved in response to advances in computing technologies.

2008 marks not only this magazine's 50th anniversary, but also my 20th year as a contributor to CACM. I was initially drawn into the ACM community to decrypt the legal theories underlying the then highly controversial software copyright "look and feel" lawsuits, which were creative, if ultimately unsuccessful, attempts to hack intellectual property law to limit competitive imitations. Apple Computer brought one such suit against Microsoft Corp., and Lotus Development Corp. another against Paperback Software.

Back then, things didn't look so good for Microsoft or Paperback because a widely cited appellate court decision from 1986, *Whelan Associates v. Jaslow Dental Labs*, had opined that computer programs should enjoy a broad scope of copyright protection, including for program structure, sequence, and organization ("SSO") and for the look and feel of their user interfaces and seemingly for program behavior.

Things looked even worse for Microsoft in 1990 after Paperback lost at the trial court level and ran out

Contrary to the dire predictions of some who favored a *Whelan*-like approach, the software industry has flourished without broad copyright protection.

of money before it could pursue an appeal. The *Paperback* decision's endorsement of *Whelan* was not only another arrow in Apple's quiver, but it bolstered Lotus' confidence that it could win subsequent lawsuits, first, against Borland International and then perhaps against Microsoft. The Borland and Microsoft spreadsheet programs allowed users to execute macros built in Lotus 1-2-3, which necessarily involved reproducing the Lotus command hierarchy so successfully challenged in *Paperback*.

1992 was a turning point in software copyright law. First, *Computer Associates v. Altai* discredited *Whelan* and rejected its analysis, holding that even if interface specifications were program "SSO," copyright did not protect them because of their importance to achieving interoperability. Second, *Sega Enterprises v. Accolade* held that making copies of program code for a legitimate purpose such as getting access to interface information in order to make a compatible program was a fair and non-infringing use of the copyrighted code. Third, a judge rejected Apple's *Whelan*-inspired theory that the look and feel of Microsoft's graphical user interface was too similar to that of Apple's Macintosh interface.

Although Lotus initially won an important round in its look and feel lawsuit against Borland in 1992, three years later an appellate court rejected Lotus's look and feel and "SSO" claims against Borland. Although Lotus appealed further to the U.S. Supreme Court, it could not persuade the Court to reinstate its victory, and finally *Whelan* lost its potency.

In retrospect, one can discern that over the course of a decade, judges managed to hack new legal doctrines out of the policy ether so that copyright law could be applied to computer programs in a competitively balanced way. Contrary to the dire predictions of some who favored a *Whelan*-like approach, the software industry has flourished without broad copyright protection.

Once *Altai* displaced *Whelan*, it became clear that copyright protected program code and expressive aspects of screen displays, but not much else. This was because *Altai* directed that functional design elements of programs had to be "filtered out" before assessing whether infringement had occurred.

The increasing "thinness" of program copyrights may have catalyzed a concomitant boom in patent applications for software innovations starting in the mid-1990s. Unfortunately, many software patents are of questionable validity owing in part to inadequacies in the patent office's prior art databases and the low standards (until very recently) for judging the nonobviousness of claimed inventions.

Hence, courts have once again been called upon to hack intellectual property law to make it appropriately responsive to the needs of the software industry, this time on the patent side. In the past few years, the U.S. Supreme Court has performed some impressive hacks. It rejected the Federal Circuit's inflexibly harsh standards for issuing injunctions in *eBay v. MercExchange*. Four members of the Court recognized that the Federal Circuit's approach had given patent trolls too much leverage over makers of complex systems technologies, such as software, only one small part of which might infringe a patent.

The Court also rejected the Federal Circuit's erroneously low standard for judging the nonobviousness of claimed inventions in the *KSR v. Teleflex* case. In addition, it agreed with Microsoft that shipping a master disk from the U.S. to an overseas destination should not give rise to damage awards in U.S. courts for acts performed abroad that would infringe if done in the U.S.

But the Court alone cannot achieve all of the needed patent reforms. Congress should pass legislation to create a new post-grant review procedure to

provide a lower-cost way to challenge the validity of questionable patents.

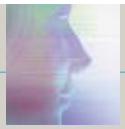
Whelan has not been the only "bad" IP hack in the past two decades. Another one was *MAI v. Peak*, which opined that temporary copies made in RAM when a computer is booted are reproductions of copyrighted software that can give rise to infringement claims if the machine was turned on by an unlicensed person.

But good hacks have been more common. *Religious Technology Center v. Netcom*, for instance, rejected an *MAI v. Peak*-like theory of liability against an Internet access provider. The judge decided that an Internet access provider should not be held liable for infringing copies of user postings on Usenet because copyright liability should be imposed only for volitional acts, not for automatic copies made by servers.

Another good hack was the Supreme Court's decision in *MGM v. Grokster*, which retained the *Sony* safe harbor for technologies having substantial noninfringing uses and held that peer-to-peer file-sharing firms should only be liable for infringement if they have induced users to infringe.

There is no way to foretell hacking of intellectual property law will be necessary to further adapt it in response to advances in computing technologies. More innovation is surely on the way—along with more lawsuits. Thus, a third decade of "Legally Speaking" columns may still be needed to translate what these lawsuits will mean for CACM readers. C

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



BY STEPHEN B. JENKINS

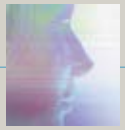
ODE TO CODE

Much have I travell'd in the realms of code,
And many goodly programs have I seen.
I've voyaged far to conferences umpteen,
Attending to the wisdom there bestowed.

Yet as I've moved along the winding road
Of my career (a journey not serene),
Only one source of knowledge has there been
Of worth enough to prompt of me an ode.

Communications has for 50 years,
Been there to help each of us on our way,
By giving us the writings of our peers,
And telling us the things they had to say.
So as the start of its sixth decade nears
Please join me wishing it "Happy Birthday."

Stephen B. Jenkins (Stephen.Jenkins@nrc-cnrc.gc.ca) is the senior programmer/analyst at the Aerodynamics Laboratory of the Institute for Aerospace Research, National Research Council Canada.



BY GUL AGHA

COMPUTING IN PERVASIVE CYBERSPACE

Freed from the temporal constraints of hardware, software could be the ultimate cyberorganism—a mind taking a body as needed to fulfill a particular function or mission.

The idea of software as a program representing a sequence of instructions on a von Neumann architecture is no longer tenable. In an article in *Communications* almost two decades ago [1], I advocated building software systems by composing concurrent objects, or actors. Actors reflect several key characteristics. For example, they are distributed, autonomous objects that interact by sending each other messages. They have unique names that are not tied to their current location, thus facilitating mobility. And new actors may be dynamically created, thus allowing new services to be added to a system. With the growth of P2P computing, Web services, networks of embedded computers, and multicore architectures, programming using the actor model is inevitable; witness the increasing use of programming languages (such as Erlang, E, SALSA, Scala, and Ptolemy) and the various implementations of software agents.

The notion of a sequential algorithm has a venerable tradition. The ninth-century Persian rationalist philosopher and mathematician Al-Khwarzimi is credited with introducing the concept of a sequence of instructions to compute a function (later termed an algorithm in his honor). But it wasn't until the mid-20th century that the discipline of computing took root, inspired by Alan Turing's representation of programs as data and by Alonzo Church's theory of computation as something that can be carried out on a Turing machine. CACM has covered the development of computer science almost since this inception.

Computing has been morphing ever since. Initially developed as a flexible calculator for scientific problems, computers have successively become an arbiter and recorder of business transactions, a reasoning engine carrying out symbolic computations, a laboratory for running simulations, and a vehicle for social networking and entertainment. At the same time,

their speed has increased more than 10-million-fold, and they have been interconnected through ever-larger bandwidths. Amazingly, our concept of a program as an implementation of a sequential algorithm has remained the same. While the shift to actor-oriented computing is overdue, we need to start thinking of software systems beyond the composition of actors.

Actors residing on a single node, even if mobile, cannot address the inevitable challenges that will characterize the future of computing. Computers are increasingly ubiquitous and embedded in all sorts of devices, from refrigerators and thermostats to automobiles and wheelchairs. The next logical step is for these embedded computers to be networked, not just in relatively localized sensor networks but through their connectivity to more powerful computers (base stations) to be globally networked. Such networking will result in a cyberspace that parallels physical space and is as pervasive as physical objects in the real world.

Pervasive cyberspace will consist of computers with heterogeneous architectures, a variety of sensing and actuation capabilities, and security and privacy profiles, many of which turn mechanical things (such as chairs, desks, windows, and walls) into smart devices. The availability of services and resources on these computers will be dynamic.

Explicit programming of such an evolving system is not feasible; elements in the pervasive cyberspace will continue to be added and removed, and architectures and policies will keep changing. Moreover, user requirements will continue to evolve. In such an environment, programmers cannot possibly know where all the resources are or will be or how and when they may be accessed. Thus software in a complex environment cannot consist of preprogrammed sequences of actions.

It is not that programming as such will be obsolete; certain local behaviors of these computers will continue to be programmed, albeit more-or-less automatically from high-level user-friendly interfaces and domain-specific libraries, written in

A cyberorganism may freely split and join other organisms, its parts may be distributed, and the parts may be destroyed and regenerate.

inherently parallel actor languages that enable the use of multicore processors. However, to facilitate more complex and interesting services, software must be able to endure and adapt. Thus, the concept of an actor as an ephemeral process tied to a single computer at a time is not sufficient to execute such services.

Endurance and adaptation are common traits in natural systems that require a relatively "long" life cycle depending on their ability to adapt through feedback mechanisms provided by the environment. Natural selection uses such feedback to change the programming of organisms.

Organisms with mobility and more complex functions (like animals in nature) use environmental feedback to learn complex new tasks. Software must also learn to purposefully navigate its environment. As such, it must be more like an animal: mobile and able to forage for resources, learn from the environment, evolve, and provide services. One can imagine cyberorganisms [2] roaming the pervasive cyberspace, gaining "wealth" by providing services, negotiating with computers to sense the environment, and computing, actuating, and learning (self-modifying) from the results of their actions.

This is not to argue that software will be entirely analogous to biological hardware—even if we were to create biological computers, as we will eventually. An innate difference between organisms and software will persist: Software cannot directly sense, compute, or affect its environment; rather, it depends on the availability of computer nodes and networks. It is like a Platonic ideal of mind with a life independent of a physical body, albeit one that may move from one body to another, adapt itself or modify how the body it controls may operate.

However, this metaphor also suggests the possibility of a different sort of resilience: a cyberorganism need not be bounded in space and time like its biological counterparts. Rather, it may freely split and join other organisms, its parts may be distributed, and the parts may be destroyed and possibly regenerate.

In pervasive cyberspace, no central authority exercises complete control over the actions of cyberorganisms. This does not mean that control mechanisms will be entirely absent. We can foresee a pervasive cyberspace with many autonomous monitoring agents and mechanisms that limit access. Some mechanisms will affect systems as a whole, others the interaction among nominally independent systems. Some of these monitoring-and-control mechanisms will affect macro-level properties of systems, much as a central bank influences the emergent properties of a national economy by controlling key interest rates.

We are at the threshold of an entirely new science of computing, one that will be inspired by the biological metaphor, not by the notion of algorithm. Reporting this scientific evolution, the next 50 years of CACM should make even more exciting reading than the previous 50 years. C

References

1. Agha, G. Concurrent object-oriented programming. *Commun. ACM 33,* 9 (Sept. 1990), 125–141.
2. Jamali, N., Thati, P., and Agha, G. An actor-based architecture for customizing and controlling agent ensembles. *IEEE Intelligent Systems 14,* 2 (Apr. 1999), 38–44.

Gul Agha (agha@cs.uiuc.edu) is Director of the Open Systems Laboratory and a professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He was Editor-in-Chief of *ACM Computing Surveys* 1999–2007.

BY GREGORY CONTI

COULD GOOGLING TAKE DOWN A PRESIDENT?

Everything we do online is known and knowable and can be combined with everything else that is known and knowable.

In the August 1984 *Communications,* Ken Thompson taught us to question our notion of trust, recognizing that even our most carefully crafted code might not generate trustworthy executable programs if the compiler is compromised [5]. Looking to the future, however, I realize Thompson didn't go far enough. Today, we must question our trust in all aspects of the information environment, including online companies and even the infrastructure of the Internet. We live in an era of rampant data disclosure and ubiquitous implied trust—two factors that will come to haunt us in the near future.

We disclose sensitive and ultimately personally identifiable information to our Internet service providers (ISPs) and favorite online organizations of every type and purpose each time we sit down at the computer. Don't believe me? Imagine if Google, Yahoo, or MSN aggregated and mined every search query emanating from your corporate IP space and every email containing your corporate domain name. Strategic plans, blackmailable material, health concerns, and social networks would all emerge. We are placing unprecedented power in the hands of the most popular online companies and ISPs, along with thousands of others, and there will come a time when it will be difficult or impossible to wrest back that power.

Could Googling take down a president, prime minister, congressman, or senator? The question is provocative but worth considering as we face the near future of trust and privacy. Googling[1] is an integral

[1]By Googling I mean the full spectrum of free online tools and services (such as search, mapping, email, Web-based word processing and calendaring).

part of the Internet fabric. Approximately one billion Internet users worldwide rely on networked technology to provide information and interconnection for all aspects of their lives, both personal and professional. Everything from our physical location, to what we think, to who we know can all be found in this data. Despite the best intentions of those doing the collecting or communicating, it is impossible to guarantee it will stay private or not be used for some malicious purpose. As an example, AOL disclosed, in August 2006, the search queries of some 657,000 of its users that contained sensitive and personally identifying information [1]. This incident only hints at the risks the world's most powerful leaders, as well as ordinary citizens, face when using myriad "free" tools (such as search, email, mapping, news, word processing, calendaring, and blog hosting). Free online services aren't really free; we pay for them with micropayments in the form of personal information [3].

One billion users, while a very large number, represents less than 18% of the global population and just a fraction of those who will turn to the Internet in the future. Although some progress has been made, these most sensitive of our hopes, dreams, and intentions [2] are routinely passed to online companies that scrupulously log and retain our revelations, sometimes indefinitely, where they are data-mined to allow customized advertising and help improve our online experience. Encryption offers little help, as online companies are a trusted party in the communication. Your computer and its Internet connections accelerate the loss of privacy. Counterintuitively, the more usable a given online application, the worse it is in terms of our personal privacy. Online companies are not the only ones with access to this information. It also flows across the networks of our ISPs, which have the power to collect, and even alter, practically every bit we send and receive. The information visible to online companies and the ISPs is largely the same; only the network vantage point is different.

In most instances of online navigation and interaction, it would be prudent to assume that these disclosures are never discarded. Once a disclosure is made, it can never be undone. At best, we must trust our ISPs and online organizations to eventually discard the information. At the same time, network neutrality is under attack. We cannot assume the information we receive is what the information provider actually sent.

In some ways, trust is increasingly irrelevant, because, if we are to be members of the Internet-enabled society, we have no other option but to rely on the powerful tools we have at our disposal (such as those provided by major search engines). Like rats forced to endure electric shocks to acquire food, we must use these tools to acquire information and communicate. The implications of data disclosure and retention are profound, including corporate and law-enforcement abuses and identity theft, as well as second- and third-order effects impossible to predict. Those of us who are aware of the risks already self-censor our activities, even as we continue to indulge them.

Those of us who are aware of the risks already self-censor our activities, even as we continue to indulge them.

What is most worrisome is less that the data is being collected at any given moment and more how it will be used (and abused) in the future. Future advances in data mining, profiling, and machine learning are particularly worrisome. While I don't foresee a dystopia in the near future, I do see a steady decline in individual freedoms and civil liberties. This decline is not new, dating back to at least the 1970s when large computerized databases of personal information were being formed in earnest. The pace accelerated globally in the aftermath of 9/11. Will we eventually reach equilibrium? I think not. The gravitational pull of both profit and power will continue to drive the decline.

Public outcry may have the power to stem the tide, but public opinion is fickle. Even the 2005 Sony rootkit incident, in which tainted Sony CDs were able to infect hundreds of thousands of end-user PCs, and the 2006 AOL data spill did little to penetrate the public consciousness. In one 2007 study only 16% of the participants reported being familiar with the AOL incident six months after it took place [4]. If this lack of public interest characterizes the general population, a less extreme rate of change will be unable to generate enough resistance to make a difference.

People have only a small window of experience to use as a reference. Chances are you lived through 9/11 and knew adult life before that day. You have a reference point, but when our generation is gone, few guides will be available to show how to defend our personal privacy. Those in power are loathe to relinquish or even share it. And, as the power and control this information (and its data-mined results) provides over hundreds of millions of citizens is seductive, corruption is inevitable. Action is critical, before it is too late to forestall individuals from losing control of their own data and perhaps even of their digital identities.

I don't want to live my life inside a Faraday cage and abandon the Internet. To do so would force me to withdraw from modern society. The future I foresee isn't guaranteed; each of us has the innate ability to influence the trajectory of technology development and use. The public is unaware, apathetic, or sees no other option than the status quo. But each of us is able to change it. As the world's leading technologists, we have the power to seek and find equitable solutions that would protect our privacy, increase our trust, and still allow online businesses, social interaction, and network providers to innovate and flourish.

In the future, Googling could indeed take down a president, yield a cure for cancer, and ruin or enrich our lives. We have to live with the past decade's worth of disclosures, but promising solutions are on the horizon. Whether they include paying for privacy, better tools for self-monitoring online activity, anonymous browsing, informed law-making, privacy-protecting corporate policy, increased user awareness, or something yet to be discovered, the solution is up to each of us. C

References

1. Barbaro, M. and Zeller, T. A face is exposed for AOL searcher no. 4417749. *The New York Times* (Aug, 9, 2006); www.nytimes.com/2006/08/09/technology/09aol.html?ex=1312776000en=f6f61949c6da4d38ei=5090.
2. Battelle, J. The database of intentions. John Battelle's Searchblog (Nov. 13, 2003); battellemedia.com/archives/000063.php.
3. Conti, G. Googling considered harmful. In *Proceedings of the New Security Paradigms Workshop* (Schloss Dagstuhl, Germany, Sept. 19–22). ACM Press, New York, 2006, 67–76.
4. Conti, G. and Sobiesk, E. An honest man has nothing to fear: User perceptions on Web-based information disclosure. In *Proceedings of the Symposium on Usable Privacy and Security* (Pittsburgh, July 18–20). ACM Press, New York, 2007, 112–121.
5. Thompson, K. Reflections on trusting trust. *Commun. ACM 27,* 8 (Aug. 1984), 761–763.

Gregory Conti (conti@acm.org) is Director of the Information Technology and Operations Center and an Academy Professor of Computer Science at the United States Military Academy, West Point, NY.

The views expressed here are those of the author and do not reflect the official policy or position of the United States Military Academy, the Department of the Army, the Department of Defense, or the U.S. Government.

BY JON CROWCROFT

TOWARD A NETWORK ARCHITECTURE THAT DOES EVERYTHING

In the same way light propagates through a medium, analogous wave-particle principles could help model communications through the future Internet architecture.

Here's a new way (I've liberally adapted from physics) to define the future network paradigm: Use the notion of wave-particle duality to view a network with swarms of coded content as the dual of packets. The wave model maps all the way down the metaphor food chain to the analog level but should be seen mainly as an analogy that works like this: First, new sources of content introduce material at multiple places in the network (including through sensor, video, and audio input), representing the start of a new wave of network traffic. The content spreads by matching agent and user subscriptions/interests to content descriptions at rendezvous points throughout the network. The analogy is also likely to go wrong in interesting ways. I hope we'll be able to use them to inspire us to come up with a new unified network architecture, sharing it in future issues of *Communications.*

One temptation computer scientists are known to indulge is to spin grand unified theories, possibly due to an innate inferiority complex when looking over the fence at physics (or perhaps because some of us started life as physicists). Whatever the reason, it shows up in networking as a desire to unify all communications under a single all-inclusive, all-welcoming design, system, or architecture. In telecommunications networks, historically successful for much longer than the Internet, the paradigm is circuit switching. Meanwhile, broadcast networks (first radio, later TV) have been around for the past century. Multiplexed, isolated circuits still dominate; for example, more than 2.5 billion cell phones in the world today operate this way, despite the growth and promise of voice over IP on the Internet. Two side effects of this design choice are that calls are billable and the use of resources is quantifiable.

What about the future Internet? Many research programs have been proposed, including the Future

Internet Design project in the U.S. and the Future Internet Research and Experimentation project in the European Union, as well as similar efforts in China, Japan, and Korea; it's more than talk. Here, I propose that we take this opportunity to think more deeply about the fundamentals of communications systems in a variety of disruptive ways to try to escape the intellectual rut we may otherwise get stuck in.

The Internet is built on the packet-switching paradigm, famously co-devised in parallel by Paul Baran and Donald Davies in the early 1970s, replacing the metaphor of electrical circuits and pipes with the idea of statistical multiplexing. Thanks to statistical multiplexing, resource sharing is more flexible than the fixed partitioning used in previous networks, and thanks to buffers, bursty sources of data can take advantage of idle lines in the network, leading to potential efficiency gains. A debate raged in the late 1970s and early 1980s between those favoring the "virtual circuit" and those favoring the "datagram" model of how to build a packet-switching-based communications system. It

The data is in some sense a shifting interference pattern that emerges from the mixing and merging of all sources.

The notion of a wave is optimized for resilience through massive scale, not for local efficiency.

turns out that the idea of a "flow" in the Internet is not very different from a virtual circuit; indeed, with so many stateful middle boxes (such as network address translation, firewalls, proxies, and multiprotocol-label-switching switched/routers), one can now say that the whole debate was futile.

However, the future needs of networks will be different from the simple circuit or packet system that has dominated for the past century, as society shifts from its historical patterns of work and entertainment. The vast majority of network content is pull/interest-based; economics [1] argues for making the "long tail" of content available at low transaction/network costs to large numbers of small groups of only the most interested recipients.

The decrease in the need for synchronization in remote activities for video, audio, and static content argues that networks, including the Internet, be optimized for nonconcurrent use. On the other hand, people want low latency, which argues for nearby or local copies of all content. Thus, we might talk about "asynchronization of multicast" and commercialization of peer-to-peer communication and content sharing. Rich-value content creators would love to eliminate any intermediaries while also pushing storage and its costs to edge users.

Technology push also plays a role in Internet-based communications. Software has changed since the days of layered system design; today, we sustain reliable software built from well-contained components assembled with wrappers designed to enforce behavior. How is this relevant to a future Internet architecture? For one thing, that architecture could be more diversified, with less commonality in the nodes (hosts and routers) than we have had for the past 20 years of PCs, servers, and routers all coming from the same technology stable.

This also fits my wave-particle model of how technology is changing within networks and protocols. Recent papers, including [2], have proposed replacing the layered protocol stack with a graph or even a heap (pile) of soft protocol components. However, we can also look at the network topology itself and see that the role of nodes is changing within it. Perhaps all nodes are the equivalent of middle boxes, revisiting the old Internet idea that any component with more than one network interface can be a router. We see it in our end-user devices—in my case the Macintosh iBook I typed this essay on and the Windows smart phone in my pocket, each with at least three built-in radios and a wired network interface. When we interconnect these devices, the network communication topology is far more dynamic than any public network has been in the past.

Many of the increasingly heterogeneous "links" connecting devices are also not well characterized as "pipes"; indeed, the capacity of a volume of space containing a number of mobile devices is simply not known; some physical bounds are known, but the equivalent of a Shannon Limit in information-theory terms is not. This lack of information argues that network architects need a temporal graph model of the network. It also argues that today's architectures are unable to accommodate the resources model in a temporal graph. Simply labeling the edges of the graph with weights to represent capacity or delay does not capture the right information.

One more critical piece of technology—network coding—further muddies the effort to devise a grand unified network architecture that would maintain the wave-particle duality analogy. Practical network coding in wireless and wired networks promises to remove much of the complexity in resource management. Network coding involves the merging of packets (such as by XOR in the simplest form) transmitted along common subpaths. Network coding can be combined with redundancy for reliability.

So how might the wave-particle duality idea be applied to a network? The Internet is already dominated by swarms of network-coded content, no longer flowing from place to place but emanating like ripples on a pond from multiple points, then becoming available for local consumption. Neal Stephenson predicted this with remarkable prescience in his novel *The Diamond Age* [3]. Publication of new content is the start of a new wave. The content spreads through

the automated matching of subscriptions/interests to content descriptions. Content is coded as it moves through the nodes in the network. A snapshot of "packets" (on an edge or stored in a node) at any given point in the graph would show they contain a coded multiplex of multiple sources of data. Hence, there would be a poor fit throughout the network architecture for packets, flow-level descriptions, normal capacity assignments, and end-to-end and hop-by-hop protocols. The data is in some sense a shifting interference pattern that emerges from the mixing and merging of all sources.

Have we also unintentionally thrown out the legacy system with the new paradigm? What about person-to-person voice calls and its 21st century equivalent, real-time gaming? If we could push the idea of swarms or waves down into the network architecture, how would the architecture implement circuit-on-a-wave and IP-on-a-wave?

Network architects could do this the same way (inefficiently) they implement VoIP—through a circuit on IP. One is at liberty to run multiple legacy networks, supporting one-to-one flows using separate communications systems, especially since the networks are available already. On the other hand, how would they be supported on the wave? Perhaps through some minimalist publication-and-subscription system.

Other ways to understand this design concept are circulating in the research community. One is the data-orientated paradigm in which information is indexed by keys and retrieved by subscription. Protocols are declarative. All nodes are caches of content, indexes, and buffers. All nodes forward information while caching, in the style of mobile ad hoc, delay-tolerant, and peer-to-peer systems; these communication methods are unified in the data-oriented paradigm.

No network architect interested in developing a grand unified network architecture would be concerned with micromanaging fine-grain resources. For a network architect, efficiency is measured at the global level. Traditional activities may be maddeningly inefficient, but most content—video, audio, and sensor data—is handled with maximum efficiency. Content is also handled through multi-path, coded delivery, with good isolation and protection properties through the statistics of scaling, not by virtue of local resource reservation.

So, unlike traditional network architectural goals, the wave-particle duality model I've described here pursues a different primary goal. In it, the notion of a wave is optimized for resilience through massive scale, not for local efficiency. Moreover, it supports group communication and mobility naturally, since the rendezvous in the network between publish and consumption is dynamic, not through the coordination of end-points in the classical end-to-end approach.

The details of the wave model are likely to keep researchers busy for the next 20 years. My aim here is to get them to think outside the end-to-end communications box in order to solve the related problems, if they are indeed the right problems, or to propose a better problem statement to begin with.

One might ask many questions about future wave-particle network architecture, including: What is the role of intermediate and end-user nodes? How do they differ? Where would be the best locus for a rendezvous between publication and consumption? Would each rendezvous depend on the popularity of content and its distance from the publisher, subscriber, or mid-point. What codes should be used? How can we build optical hardware to achieve software re-coding? What role might interference in radio networks play in the wave-particle network model? How can we achieve privacy for individual users and their communications in a network that mixes data packets?

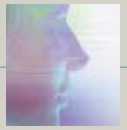
This future wave-particle duality in the Internet-based network would be more resilient to failure, noise, and attack than the current architecture where ends and intermediate nodes on a path are sitting ducks for attacks, whether deliberate or accidental. How might its architects quantify the performance of such a system? Do they need a new set of modeling tools—replacing graph theory and queuing systems—to describe it? Finally, if network control is indeed a distributed system, can the idea of peer-to-peer be used as a control plane?

I encourage you not to take my wave-particle duality analogy too seriously, especially since I am suspicious of any grand unified network model myself. But I do encourage you to use the idea to disrupt your own thinking about traditional ideas. In the end, perhaps, we will together discover that many traditional ideas in networking are fine as is, but all are still worth checking from this new perspective. C

References

1. Anderson, C. *The Long Tail: How Endless Choice Is Creating Unlimited Demand.* Random House, New York, 2006.
2. Braden, R., Faber, T., and Handley, M. From protocol stack to protocol heap: Role-based architecture. *ACM SIGCOMM Computer Communication Review 33,* 1 (Jan. 2003), 17–22.
3. Stephenson, N. *The Diamond Age: Or, a Young Lady's Illustrated Primer.* Bantam Spectra Books, New York, 1995.

Jon Crowcroft (Jon.Crowcroft@cl.cam.ac.uk) is the Marconi Professor of Communications Systems in the Computer Laboratory at the University of Cambridge, Cambridge, U.K.



BY PETER G. NEUMANN

REFLECTIONS ON COMPUTER-RELATED RISKS

Tracing the history of exposing and elucidating the wide variety of system problems and associated social implications of uses and misuses of computing technology.

Computer-related technologies have changed enormously over the years, with huge advances in processor power and storage capacity, high-speed networking, and highly distributed systems. Client-server and virtual-machine architectures seem to be simplifying implementation. Internet browsers have significantly raised the level of abstraction for attaining almost universal interoperability. Strong cryptography has become more widely available, and is becoming easier to use. Improvements in static-analysis tools and formal methods are having visible results. Many ACM members have been instrumental in some wonderful advances, and have been involved in important technological and social activities. For example, Parnas, Dijkstra, Hoare, Wirth, and many others contributed to system architectures and programming practice. We have also experienced significant advances in networking, graphics, and many other crucial areas.

Conversely, trustworthiness of operating systems and application software is generally poor, particularly with respect to critical requirements such as security, reliability, survivability, evolvability, maintainability, interoperability, and predictable upgradability. Common flaws keep recurring—buffer overflows, faulty bounds checks, and so on. Denial-of-service attacks are easily created and deployed, but largely lacking adequate defenses. Strong cryptography is difficult to embed securely in systems and applications. Software engineering is still more of an ideal concept rather than a disciplined practice; its principled precepts seem to be widely ignored. The Internet has amplified the risks, and seems to encourage various spams, scams, and spoofs. Trustworthiness and particularly security are

often not adequately recognized as essential elements—especially in system architectures, system development, and in curricula. The attackers seem to be gaining faster than the defenders. The current state of the practice in the use of computer systems in elections is particularly appalling; the standards are weak, and the bar is set way below the financial sector and even gambling machines. Vulnerabilities in our critical infrastructures are equally worrisome. High-assurance multilevel security is still more or less a dream, although its practical existence in mainstream systems would provide possibilities that do not exist today.

These and many more subjects have been considered in ACM's SIGSOFT *Software Engineering Notes* (SEN) (which I created in 1976 and edited until Will Tracz took over in 1994; www.sigsoft.org/SEN/), the ACM Risks Forum (since 1985; www.risks.org), and CACM's "Inside Risks" columns (since July 1990; www.csl.sri.com/neumann/insiderisks.html). (My book *Computer-Related Risks*, published in 1995, is still basically sound despite its age, because many things have not fundamentally changed.) Thus, it seems useful to provide some background that might not be familiar especially to younger ACM members, and to consider what lessons might be learned therefrom.

SEN has served as an outlet for discussions of systems that did not work as expected, as well as how such problems might be avoided. But that was perhaps only preaching to the converted. For several years, SEN included an annual updated list of Illustrative Risks to the Public in the Use of Computers—until the list became too long and became searchable online (see www.csl.sri.com/neumann/illustrative.html). With the ever-increasing volume of salient RISKS cases, I am less inclined to keep the index current. Besides, Lindsay Marshall has provided a nice searchable Web site at Newcastle University for

Risky problems are as great today as they were when we first set out to expose and eradicate them.

the complete RISKS archives (risks.org).

Over those early years, there was considerable debate within the ACM Council about ACM's role in representing real-world concerns regarding the use of computers. The discussions within the Council that inspired the establishment of the ACM Risks Forum are described at length in the message from ACM's president at the time, Adele Goldberg, in the February 1985 issue of CACM. This was placed under the aegis of the ACM Committee on Computers and Public Policy (CCPP), the chairmanship of which I then inherited from Dan McCracken. ACM thereby demonstrated a genuine recognition of the importance of the social implications of our technologies.

The first RISKS issue on August 1, 1985 (see catless.ncl.ac.uk/Risks/1.01.html) includes a summary of Adele Goldberg's message with an excerpt of the charter, an agenda for the future, a summary of some of the incidents known at the time culled from SEN (which grew into the Illustrative Risks index), items on the strategic defense initiative and Dave Parnas's resignation from the antimissile defense advisory group, a pointer to Herb Lin's analysis of that software, a minireview by Peter Denning, and a note from Jim Horning.

Five years after that, CACM Editor-in-Chief Peter Denning and others urged me to establish the monthly column that became "Inside Risks." I am enormously indebted to the members of CCPP—which then included Denning, Parnas, Horning, Nancy Leveson, Jerry Saltzer, and others—who have served as an astute expert review panel for each succeeding would-be column and provided wise counsel on other issues as well.

The overwhelming conclusion from this body of material is that the risky problems are as great today as they were when we first set out to expose and eradicate them. Although the prevention mechanisms have improved somewhat, it is evident that we have not been advancing sufficiently rapidly in the development of mass-marketplace systems and custom applications that are sufficiently trustworthy—despite the tangible gains and research advances I noted in the first paragraph of this essay. Worse yet, various factors have outpaced those mechanisms, including increased complexity of systems, increased worldwide dependence on information technology and the ever-growing Internet, increasingly critical applications to which that technology is being entrusted, the general ease with which antisocial acts can be committed, and the ubiquity of potential attackers. Thus, we seem to be falling farther behind as time goes by. In particular, the huge expansion in the scope and pervasiveness of the Internet is creating many challenges for our community.

One of the biggest challenges for ACM members and for the computer community as a whole is bridging the gap between research and development, and the gap between theory and practice. Clearly, we need to devote greater attention to improving development practices.

In its first 50 years, CACM has been a useful product of the Association for Computing Machinery. However, in the next 50 years, the ACM needs to become—both in spirit and in reality—something more like the Association for Computing Methods or perhaps Methodologists, stressing the vital role of people in the urgent pursuit of transforming computer system development into a true engineering discipline that makes optimal use of the advances of the past 50 years in the context of critical system applications that use the resulting systems wisely. In particular, dramatic changes are needed in developing trustworthy systems that are explicitly designed for human usability for all users, and that encourage well-informed people to take on appropriate responsibilities in environments in which it is clearly unwise to trust technology blindly. For example, see the recommendations of the National Research Council study relating to secure systems and networks, summarized in the October 2007 CACM "Inside Risks" column and the columns relating to the needs for total-system understanding, education, and consistent application of good system-oriented principles.

In 1954, Norbert Wiener wrote about the use of human beings in the context of what he foresaw as the future of computer systems. In 2008, we need to remember that although ACM seeks to improve computer-related technologies and their applications, the purpose of that technology is ultimately to improve the quality of the life for everyone on our planet. C

Peter G. Neumann (neumann@csl.sri.com) is Principal Scientist at SRI International's Computer Science Lab in Menlo Park, CA. He also chairs the ACM Committee on Computers and Public Policy and is the moderator of the ACM Risks Forum.

Call For Papers
Sept. 23-26, *2008*
Beijing, China

IEEE International Conference on Services Computing (SCC 2008)

July 8 - 11, 2008, Hawaii, USA
http://conferences.computer.org/scc/2008

Services Computing has become a foundational discipline that covers the science and technology of services innovation research that leverages IT and computing technology to model, create, and manage business solutions, scientific applications, as well as modernized services. SCC 2008 continues to bridge the gap between Services Computing and business models with an emerging suite of ground-breaking technology that includes Service Oriented Architecture (SOA), business process integration and management, services engineering and utility/grid computing. The theme of SCC 2008 is "Services: Business, Technology, and Application."

SCC 2008 seeks orginal, unpublished research and industry/application papers in the following major tracks:

- Foundations of Services Computing
- Business Models and Operations in Services Industry
- Business Process Management and Integration
- SOA Toos, Solutions, and Services

SCC 2008 and SERVICES 2008 (Part A) Important Dates
Paper Submission Due : 1/28/2008
Decision Notification : 3/25/2008
Camera-Ready Due : 4/18/2008

Sponsored Journals for ICWS & SCC 2008
Int. Journal of Web Services Research (JWSR) (SCI)
Int. J of Business Process Integration & Management (IJBPIM)
Int. Journal of Grid &Utility Computing (IJGUC)
IEEE IT Professional Magazine
IEEE Transactions on Services Computing (TSC)

IEEE International Conference on Web Services (ICWS 2008)

September 23 - 26, 2008, Beijing, China
http://www.icws.org

ICWS has been a prime international forum for both researchers and industry practitioners to exchange the latest fundamental advances in the state of the art and practice of Web services. ICWS also aims to identify emerging research topics and define the future of Web services. Over the past six years ICWS has grown steadily attracting over 250 participants on a regular basis.

ICWS 2008 will continue to feature research and industry/application papers with a wide range of topics, focusing on the infrastructure of SOA and various aspects of IT services.

Call For Papers

ICWS 2008 seeks original, unpublished papers reporting new work in all aspects of Web Services (WS). Topics of interest include but are not limited to:

- Extended WS Specifications
- WS Publishing & Discovery
- WS Composition & Integration
- WS Testing
- Security Challenges for WS
- QoS in WS
- Web 2.0 and Web X.0 Concepts in WS
- Semantic WS
- WS-based Mobile Computing
- WS Standards and Implementation
- WS Modeling
- Relationship Binding, and Tooling

ICWS 2008 and SERVICES 2008 (Part B) Important Dates
Paper Submission Due : 4/7/2008
Decision Notification : 5/26/2008
Camera-Ready Due : 6/27/2008

IEEE Congress on Services (SERVICES 2008)

http://conferences.computer.org/services/2008
Part A: Jul. 8-11, 2008, Hawaii, USA
Part B: Sep. 23-26, 2008, Beijing,China

The two flagship conferences, ICWS 2008 & SCC 2008, are strategically part of SERVICES 2008. While ICWS & SCC will continue to focus on high-quality research and industry paper sessions, the theme of SERVICES is to support them. SERVICES 2008 will include the following key events:

Services University, 2008 IEEE Services Computing Contest, 2008 IEEE SOA Industry Summit, 2008 IEEE Services Computing Workshops (SCW), 2008 IEEE Ph.D. Student Symposium on Services Computing, 2008 IEEE Symposium on SOA Standards, Demos and Exhibits, Job Fair, IEEE Body of Knowledge (BoK) on Services Computing.

All papers reviewed and accepted by SERVICES 2008 events in Hawaii and Beijing will be published in the Proceedings of 2008 IEEE Congress on Services (SERVICES 2008).

ICWS 2008, SCC 2008, and SERVICES 2008 Conference Proceedings will be publihsed by IEEE Computer Society Press. They will be included in EI Compendex (ICWS 2003-2007, SCC 2004-2007, SERVICES 2007) & SCI (in Plan)

Based on the best practice at SERVICES 2007, all papers/presentations (To be captured in RSS/PodCast/Video formats by ICWS/SCC/SERVICES 2008) will be contributed to the *public* IEEE Body of Knowledge (BoK) on Services Computing Portal
http://www.servicescomputing.tv

Sponsored by The Technical Committee on Services Computing
http://tab.computer.org/tcsc
Contact: ZHANGLJ@IEEE.ORG

Call For Papers
July 8-11, *2008*
Hawaii, USA

BY JASON LEIGH AND MAXINE D. BROWN

CYBER-COMMONS: MERGING REAL AND VIRTUAL WORLDS

Cyber-mashups of very large data sets let users explore, analyze, and comprehend the science behind the information being streamed.

As longtime *Communications* subscribers, as well as authors of articles and contributors to special sections, notably "Blueprint for the Future of High-Performance Computing" (November 1997) and "Blueprint for the Future of High-Performance Networking" (November 2003), we appreciate the opportunity to learn from and share with the ACM community. Over the next 50 years, as computing and networking technologies advance, we'll have to be able to harness the power of technology to address challenges involving biodiversity, the environment, medicine, energy, and more. That's why we're creating visualization and collaboration user interfaces to enable global virtual organizations to work together. Happy Birthday, CACM, and many more.

As educators, researchers, and specialists in networked visualization, virtual reality, and collaboration technologies, we are fortunate to work at an institution—the Electronic Visualization Laboratory at the University of Illinois at Chicago—with access to some of the most advanced "cyberinfrastructure" in the world. That term, coined by the National Science Foundation, refers to high-performance computing and communications environments that integrate data, computers, and people on a global scale in order to stimulate scientific exploration, theories, and knowledge.

Like most university educators today, the learning environments we provide must motivate our students to excel and prepare them to qualify for careers in the global work force. As technologists, we must harness the power of emerging technologies (such as petascale computing, exabyte data stores, and terabit networks). As researchers, we must create the virtual organizations, hardware, software, and human-interface models behind the cyberinfrastructure for data-intensive scientific research and collaboration.

FIGURE 1. STUDENTS AND FACULTY GATHER IN THE CYBER-COMMONS FOR WEEKLY TECHNICAL MEETINGS.

For the past five years we have been part of the NSF-funded OptIPuter project [5], developing advanced cyberinfrastructure to enable scientists to interactively visualize, analyze, and correlate massive amounts of data from multiple storage sites connected through optical networks. One major result has been the OptIPortal, a 21st-century "personal computer" consisting of a tiled display wall connected to a computer cluster connected to multi-gigabit national and international networks. The OptIPortal runs the Scalable Adaptive Graphics Environment, software we've developed to serve as a cyber-mashup, enabling collaborators to simultaneously run applications on local or remote clusters. Discussing and analyzing the science behind the information being streamed, remote colleagues access and view multiple ultra-high-resolution visualizations, participate in high-definition videoconferencing calls, browse the Web, or show PowerPoint presentations.

The OptIPuter and its OptIPortals provide scientists and students better technologies in the laboratory and classroom than they might currently have at home. To make them useful and useable, we've created the Cyber-Commons, a community resource openly accessible to our faculty and students (see Figure 1).

David Gelernter, in his prescient 1992 book *Mirror Worlds* [3], wrote "A Mirror World is some huge institution's moving, true-to-life mirror image trapped inside a computer—where you can see and grasp it whole. The thick, dense, busy subworld that encompasses you is also, now, an object in your hands... This software technology, in combination with high-speed parallel computers and computer networks, makes it possible to envision enormous, intelligent information reservoirs linking libraries and databases across the country or the world... The Mirror World is a wholeness-enhancing instrument; it is the sort of instrument that modern life demands. It is an instrument that you (almost literally) look through, as through a telescope, to

FIGURE 2. FUTURE CYBER-COMMONS ENVIRONMENT.

see and grasp the nature of the organizational world that surrounds you."

Cyber-Commons instantiates *Mirror Worlds* as the telescope one uses to view and collect data from global resources but goes further, bringing people together, possibly in real time, to enable collaboration. The goal is not just to mirror the "universe in a shoebox" [3] but to enable people worldwide to work together to create and learn from the world-in-a-box. *Mirror Worlds* went beyond the notion of virtual worlds, foreseeing the existence of advanced optical networks that allow us to share real spaces and real data. While virtual worlds, like Second Life, are useful within the context of cyberinfrastructure when avatar-based systems are needed to explore simulated worlds, *Mirror Worlds* described a much more substantial environment.

In 1992, our laboratory, under the direction of Tom DeFanti and Dan Sandin, developed the CAVE virtual-reality theater [2]. Also in 1992, we networked the CAVE to supercomputers. By 1995, the CAVE was networked to people at more than a dozen sites [4]. As pointed out in [1], "Today's virtual worlds contrast sharply with the concept of total immersive VR that has long been popular with science fiction writers but has proven so difficult for computer scientists to achieve in the real world. Second Life and World of Warcraft images are restricted to the screen of an ordinary computer monitor, rather than filling the walls of a VR cave or binocular head-mounted display. On the one hand, this may suggest that people really do not need visually perfect VR. On the other hand, today's virtual worlds may be preparing millions of people to demand full VR in the future."

The future of real-time scientific research and collaboration is indeed in immersive environments. The requirements for a comprehensive global cyberinfrastructure are becoming more and more demanding as scientific research becomes more complex and as scientists need better collaboration and visualization technologies combined with the same digital conveniences they have at home. What domain scientists want is to interface with colleagues and data and easily mashup very large data sets in order to study and understand complex systems, from the micro to the macro scale, in space and time. To enable these cybermashups, the future Cyber-Commons must seamlessly integrate ultra-high-resolution 2D and

The future Cyber-Commons will support continuous interaction, respond to human gesture, and encourage mobility among team members and information.

autostereoscopic 3D display technologies, table displays, high-definition teleconferencing systems, laptops, and ubiquitous handheld devices, as well as support both collocated and distance knowledge discovery. The future Cyber-Commons (see Figure 2) will be a digital assistant of sorts, anticipating and enabling those who work within it, benefiting global scientific collaboratories and providing an opportunity for new computer science research.

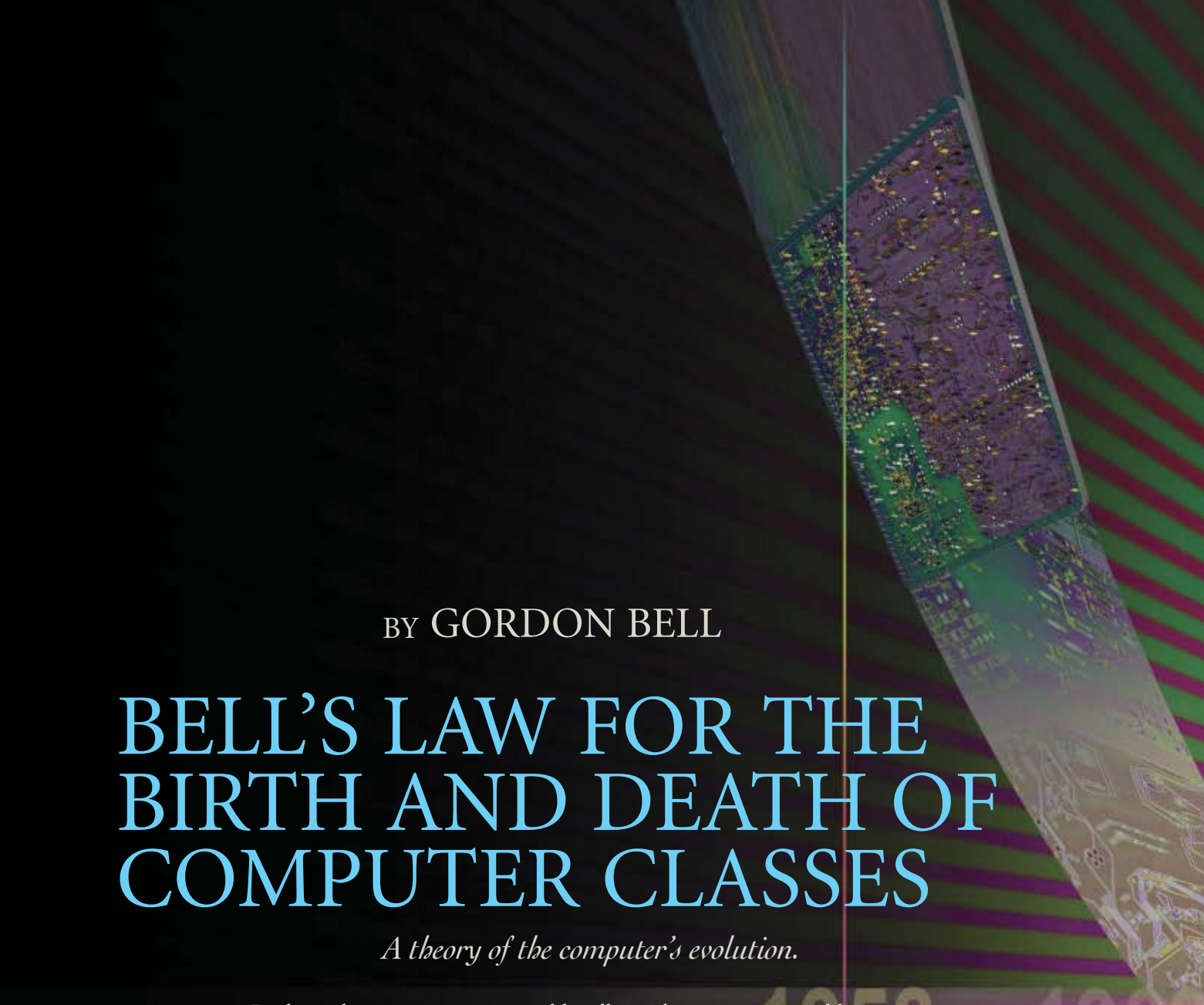
The future Cyber-Commons will support continuous interaction, respond to human gesture, and encourage mobility among team members and information. It will connect distributed teams over high-speed networks and support persistent digital artifacts, so when the power is turned off, the information on the display walls will not be lost. It will enable the seamless viewing of ultra-high-resolution 2D images and 3D stereoscopic images without special glasses. It will also support ubiquitous and intuitive interaction devices, creating a powerful and easy-to-use information-rich environment for scientific discovery and education. C

References

1. Bainbridge, W. The scientific research potential of virtual worlds. *Science 317* (July 27, 2007), 472–476.
2. Cruz-Neira, C., Sandin, D., DeFanti, T., Kenyon, R., and Hart, J. The CAVE: Audio-visual experience automatic virtual environment. *Commun. ACM 35,* 6 (June 1992), 65–72.
3. Gelernter, D. *Mirror Worlds.* Oxford University Press, 1992.
4. Leigh, J. and Johnson, A. Supporting transcontinental collaborative work in persistent virtual environments. *IEEE Computer Graphics and Applications 16,* 4 (July 1996), 47–51.
5. Smarr, L., Chien, A., DeFanti, T., Leigh, J., and Papadopoulos, P. The OptIPuter. *Commun. ACM 46,* 11 (Nov. 2003), 58–67.

Jason Leigh (spiff@evl.uic.edu) is Director of the Electronic Visualization Laboratory and an associate professor of computer science at the University of Illinois at Chicago.
Maxine D. Brown (maxine@uic.edu) is Associate Director of the Electronic Visualization Laboratory at the University of Illinois at Chicago.

This material is based on work supported by the National Science Foundation awards CNS-0420477, OCI-0441094, and OCI-0225642. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the funding agencies and companies.



BY GORDON BELL

BELL'S LAW FOR THE BIRTH AND DEATH OF COMPUTER CLASSES

A theory of the computer's evolution.

In the early 1950s, a person could walk inside a computer and by 2010 a single computer (or "cluster') with millions of processors will have expanded to the size of a building. More importantly, computers are beginning to "walk" inside of us. These ends of the computing spectrum illustrate the vast dynamic range in computing power, size, cost, and other factors for early 21st century computer classes.

A computer class is a set of computers in a particular price range with unique or similar programming environments (such as Linux, OS/360, Palm, Symbian, Windows) that support a variety of applications that communicate with people and/or other systems. A new computer class forms and approximately doubles each decade, establishing a new industry. A class may be the consequence and combination of a new platform with a new programming environment, a new network, and new interface with people and/or other information processing systems.


1988
1998
2008
2038

Bell's Law accounts for the formation, evolution, and death of computer classes based on logic technology evolution beginning with the invention of the computer and the computer industry in the first-generation, vacuum-tube computers (1950–1960), second-generation, transistor computers (1958–1970), through the invention and evolution of the third-generation Transistor-Transistor Logic (TTL) and Emitter-coupled Logic (ECL) bipolar integrated circuits (ICs) from 1965–1985. The fourth-generation MOS and CMOS ICs enabling the microprocessor (1971) represents a "break point" in the theory because it eliminated the other early, more slowly evolving technologies. Moore's Law [6] is an observation about integrated circuit semiconductor process improvements or evolution since the first IC chips, and in 2007 Moore extended the prediction for 10–15 more years, as expressed in Equation 1. The evolutionary characteristics of disks, networks, displays, user interface technologies, and programming environments will not be discussed here. However, for classes to form and evolve, all technologies must evolve in scale, size, and performance at their own—but comparable—rates [5].

$$\text{Transistors per chip} = 2^{(t-1959)} \text{ for } 1959 \le t \le 1975; 2^{16} \times 2^{(t-1975)/1.5} \text{ for } t \ge 1975. \quad (1)$$

In the first period, the mainframe, followed by minimal computers, smaller mainframes, supercomputers, and minicomputers established themselves as classes in the first and second generations and evolved with the third-generation integrated circuits circa 1965–1990. In the second or current period, with the fourth generation, marked by the single processor-on-a-chip, evolving large-scale integrated circuits (1971–present) CMOS became the single, determinant technology for establishing all computer classes. By 2010, scalable CMOS microprocessors combined into powerful, multiple processor clusters of up to one million independent computing streams are likely. Beginning in the mid-1980s, scalable systems have eliminated and replaced the previously established, more slowly evolving classes of the first period that used interconnected bipolar and ECL ICs. Simultaneously smaller, CMOS system-on-a-chip computer evolution has enabled low-cost, small form factor (SFF) or cell-phone-sized devices (CFSD); PDA, cell phone, personal audio (and video) devices (PADs), GPS, and camera convergence into a single platform will become the worldwide personal computer, circa 2010. Dust-sized chips with relatively small numbers of transistors enable the creation of ubiquitous, radio networked, implantable, sensing platforms to be part of everything and everyone as a wireless sensor network class. Field Programmable Logic Array chips with tens to hundreds of millions of cells exist as truly universal devices for building nearly anything.

Bell's Law

A computer class is a set of computers in a particular price range defined by: a programming environment such as Linux or Windows to support a variety of applications; a network; and user interface for communication with other information processing systems and people. A class establishes a horizontally structured industry composed of hardware components through operating systems, languages, application programs and unique content including databases, games, images, songs, and videos that serves a market through various distribution channels.

The universal nature of stored-program computers is such that a computer may be programmed to replicate function from another class. Hence, over time, one class may subsume or kill off another class. Computers are generally created for one or more basic information processing functions—storage, computation, communication, or control. Market demand for a class and among all classes is fairly elastic. In 2010, the number of units sold in classes will vary from tens for computers costing around $100 million to billions for SFF devices such as cell phones selling for under $100. Costs will decline by increasing volume through manufacturing learning curves (doubling the total number of units produced results in cost reduction of 10%–15%). Finally, computing resources including processing, memory, and network are fungible and can be traded off at various levels of a computing hierarchy (for example, data can be held personally or provided globally and held on the Web).

The class creation, evolution, and dissolution process can be seen in the three design styles and price trajectories and one resulting performance trajectory that threatens higher-priced classes: an established class tends to be re-implemented to maintain its price, providing increasing performance; minis or minimal-cost computer designs are created by using the technology improvements to create smaller computers used in more special ways; supercomputer design (the largest computers at a given time come into existence by competing and pushing technology to the limit to meet the unending demand for capability); and the inherent increases in performance at every class, including constant price, threaten and often subsume higher-priced classes.

All of the classes taken together that form the com-

puter and communications industry shown in Figure 1 behave generally as follows:

- Computers are born—classes come into existence through intense, competitive, entrepreneurial action over a period of two to three years to occupy a price range, through the confluence of new hardware, programming environments, networks, interfaces, applications, and distribution channels. During the formation period, two to hundreds of companies compete to establish a market position. After this formative and rapid growth period, two or three, or a dozen primary companies remain as a class reaches maturity depending on the class volume.
- A computer class, determined by a unique price range, evolves in functionality and gradually expanding price range of 10 maintains a stable market. This is followed by a similar lower-priced sub-class that expands the range another factor of five to 10. Evolution is similar to Newton's First Law (bodies maintain their motion and direction unless acted on externally). For example, the "mainframe" class was established in the early 1950s using vacuum tube technology by Univac and IBM and functionally bifurcated into commercial and scientific applications. Constant price evolution follows directly from Moore's Law whereby a given collection of chips provide more transistors and hence more performance.

  A lower entry price, similar characteristics sub-class often follows to increase the class's price range by another factor of five to 10, attracting more usage and extending the market. For example, smaller "mainframes" existed within five years after the first larger computers as sub-classes.
- CMOS semiconductor density and packaging inherently enable performance increase to support a trajectory of increasing price and function.

  Moore's Law single-chip evolution, or microprocessor computer evolution after 1971 enabled new, higher performing and more expensive classes. The initial introduction of the microprocessor at a substantially lower cost accounted for formation of the initial microcomputer that was programmed to be a calculator. This was followed by more powerful, more expensive classes forming including the home computer, PC, workstation, the shared microcomputer, and eventually every higher class. Home and personal computers are differentiated from workstations simply on "buyer"—a person versus an organization.

  The supercomputer class circa 1960 was established as the highest performance computer of the day. However, since the mid-1990s supercomputers are created by combining the largest number of high-performance microprocessor-based computers to form a single, clustered computer system in a single facility. In 2010, over a million processors will likely constitute a cluster. Geographically coupled computers including GRID computing, such as SETI@home, are outside the scope.
- Approximately every decade a new computer class forms as a new "minimal" computer either through using fewer components or use of a small fractional part of the state-of-the-art chips. For example, the hundredfold increase in component density per decade enables smaller chips, disks, and screens at the same functionality of the previous decade especially since powerful microprocessor cores (for example, the ARM) use only a few (less than 100,000) transistors versus over a billion for the largest Itanium derivatives.

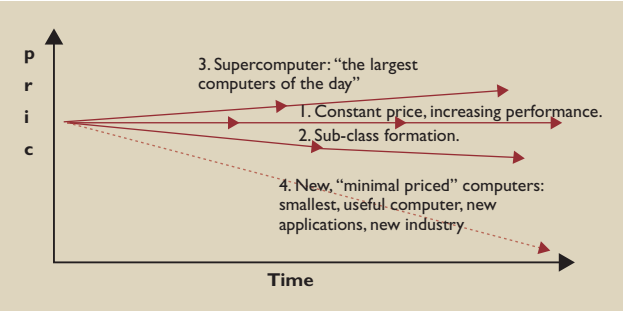


Figure 1. Evolving computer classes based on technology and design styles.

Building the smallest possible computer accounts for the creation of computers that were used by one person at a time and were forerunners of the workstation (for example, the Bendix G-15 and LGP 30 in 1955), but the first truly personal computer was the 1962 Laboratory Instrument Computer (LINC). LINC was a self-contained computer for an individual's sole use with appropriate interfacial hardware (keyboards, displays), program/data filing system, with interactive program creation and execution software. Digital Equipment's PDP-1 circa 1961, followed by the more "minimal" PDP-5 and PDP-8 established the minicomputer class [1] that was predominantly designed for embedded applications.

Systems-on-a-chip (SOCs) use a fraction of a chip for the microprocessor(s) portion or "cores" to create classes and are the basis of fixed-function devices and appliances beginning in the mid-1990s. These include cameras, cell phones, PDAs, PADs, and their convergence into a single CPSD or SFF package. This accounts for the PC's rapidly evolving microprocessor's ability to directly subsume the

1980s workstation class by 1990.

- Computer classes die or are overtaken by lower-priced, more rapidly evolving general-purpose computers as the less-expensive alternatives operating alone, combined into multiple shared memory microprocessors, and multiple computer clusters. Lower-priced platforms result in more use and substantially higher volume manufacture thereby decreasing cost while simultaneously increasing performance more rapidly than higher-priced classes.

Computers can be combined to form a single, shared-memory computer. A "multi" or multiple CMOS microprocessor, shared-memory computer [2] displaced bipolar minicomputers circa 1990 and mainframes circa 1995, and formed the basic component for supercomputers.

Scalable, multiple computers can be networked into arbitrarily large computers to form clusters that replace custom ECL and CMOS vector supercomputers beginning in the mid-1990s simply because arbitrarily large computers can be created. Clusters of multiprocessors were called constellations; clusters using low latency and proprietary networks are MPPs (massively parallel processors).

Generality *always* wins. A computer created for a particular, specialized function, such as word processing or interpreting a language, and used for a particular application is almost certain to be taken over by a faster-evolving, general-purpose computer. The computer's universality property allows any computer to take on the function of another, given sufficient memory and interfaces.

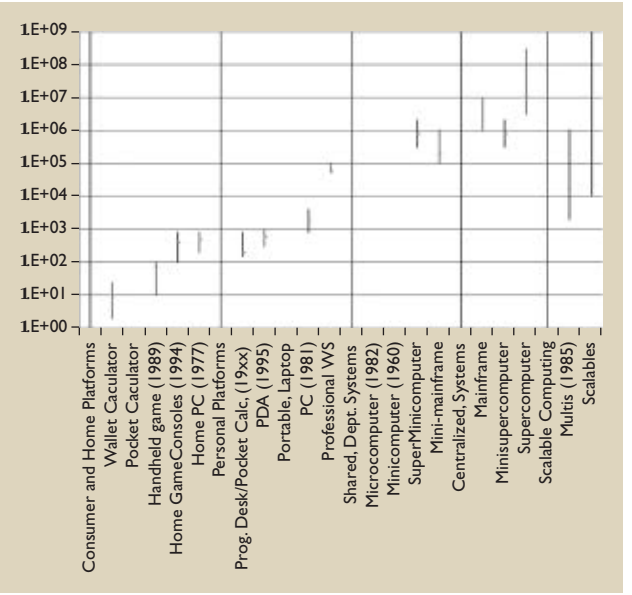


Figure 2a. Computer classes and their price range circa 2005.

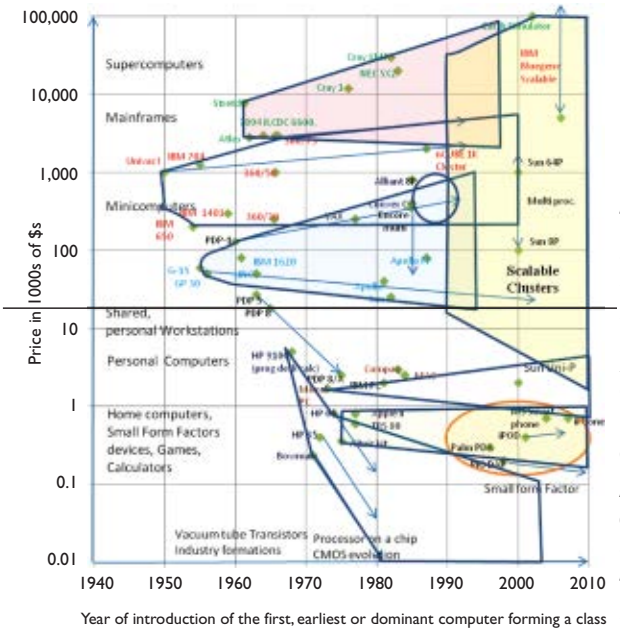


Figure 2b. Introduction price versus date of the first or early platforms to establish a computer class or lower-priced sub-class originating from the same company or industry.

SFF devices subsume personal computing functionality as they take on the communications functions of the PC (email and Web browsing), given sufficient memory and interfaces. SFF devices, TVs, or kiosks accessing supercomputers with large stores, subsume personal computing functionality. The large central stores retain personal information, photos, music, and video.

The specific characteristics of the classes account for the birth, growth, diminution, and demise of various parts of the computer and communications industry.

Overview of the Birth and Death of the Computer Classes 1951–2010

The named classes and their price range circa 2010 are given in Figure 2a. In 1986, David Nelson, the founder of Apollo computer, and I posited that the price of a computer was approximately $200 per pound [7]. Figure 2b gives the introduction price and date of the first or defining computer of a class.

Here, I will use the aspects of Bell's Law described previously and follow a timeline of the class formations beginning with the establishment of the first computer classes (mainframe, supercomputer, shared personal professional computers or workstations, and minicomputers) using vacuum tubes, transistors, and bipolar integrated circuits that continue through the mid-1990s in the first period (1951–1990). In the second period beginning in 1971, the MOS microprocessor ultimately overtook bipolar by 1990 to

establish a single line based on CMOS technology. The section is followed by the three direct and indirect effects of Moore's Law to determine classes:

- Microprocessor transistor/chip evolution circa 1971–1985 establish: calculators, home computers, personal computers, and workstations, and lower (than minicomputer) priced computers.
- "Minimal" designs establish new classes circa 1990 that use a "fraction" of the Moore number. Microsystems evolution using fractional Moore's Law-sized SOCs enable small, lower-performing, minimal PC and communication systems including PDAs, PADs, cameras, and cell phones.
- Rapidly evolving microprocessors using CMOS and a simpler RISC architecture appear as the "killer micro" circa 1985 to have the same performance as supercomputers, mainframes, mini-supercomputers, super-minicomputers, and minicomputers built from slowly evolving, low-density, custom ECL and bipolar integrated circuits. ECL survived in supercomputers the longest because of its speed and ability to drive the long transmission lines, inherent in large systems. In the end, CMOS density and faster system clocks overtook ECL by 1990.

The "killer micro" enabled by fast floating-point arithmetic subsumed workstations and minicomputers especially when combined to form the "multi" or multiple microprocessor shared memory computer circa 1985. "Multis" became the component for scalable clusters when interconnected by high-speed, low-latency networks. Clusters allow arbitrarily large computers that are limited only by customer budgets. Thus scalability allows every computer structure from a few thousand dollars to several hundred million dollars to be arranged into clusters built from the same components.

In the same fashion that killer micros subsumed all the computer classes by combining, it can be speculated that much higher volume—on the order of hundreds of millions—of SFF devices, may evolve more rapidly to subsume a large percentage of personal computing. Finally, tens of billions of dust-sized, embeddable wirelessly connected platforms that connect everything are likely to be the largest class of all enabling the state of everything to be sensed, effected, and communicated with.

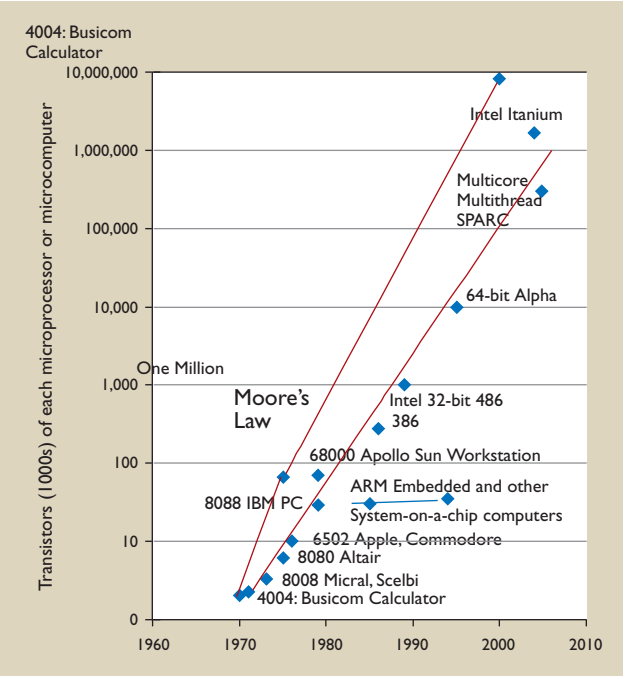


Figure 3. Moore's Law, which provides more transistors per chip per year, has resulted in creating the following computer classes: calculators, home computers, personal computers, workstations, "multis" to overtake minicomputers, and clusters using multiple core, multithreading to overtake mainframes and supercomputers.

Microprocessors circa 1971: The Evolving Force for Classes in the Second Period

Figure 3 shows the microprocessors derived directly from the growth of transistors/chips beginning in 1971. It shows the trajectory of microprocessors from a 4-bit data path through, 8-, 16-, 32-, and 64-bit data paths and address sizes. The figure shows a second path—the establishment of "minimal" computers that use less than 50 thousand transistors for the processor, leaving the remainder of the chip for memory and other functions (for example, radio, sensors, analog I/O) enabling the complete SOC. Increased performance, not shown in the figure, is a third aspect of Moore's Law that allows the "killer micro" formation to subsume all the other, high-performance classes that used more slowly evolving bipolar TTL and ECL ICs. Calculators, home computers, personal computers, and workstations were established as classes as the processor on a chip evolved to have more transistors with wide data paths and large address spaces as shown in Figure 3.

In 1971, Intel's 4004 with a 4-bit data path and ability to address 4KB was developed and programmed to be the Busicom Calculator; instead of developing a special chip as had been customary to implement calculators, a program was written for the 4004 for it to "behave" as or "emulate" a calculator.

In 1972, Intel introduced the 8008 microprocessor coming from the Datapoint terminal requirement, with a 8-bit data path and ability to access 16KB that allowed limited, programmable computers followed by more powerful 8080-based systems MITS used to introduce its Altair personal computer kit in 1975,

Over time, the performance of lesser-performing, faster-evolving products eventually overtakes the established, slowly evolving classes served by sustaining technology.

which incidentally stimulated Gates and Allen to start Microsoft. In 1977, the 16-bit 6502 microprocessor and higher-capacity memory chips enabled personal computers for use in the home or classroom built by Apple, Commodore, and Radio Shack—computers that sold in the tens of millions because people bought them to use at home versus corporate buyers. By 1979, the VisiCalc spreadsheet ran on the Apple II establishing it as a "killer app" for personal computers in a work environment. Thus, the trajectory went from a 4-bit data path and limited address space to a 16-bit data path with the ability to access 64KB of memory. This also demonstrates the importance of physical address as an architectural limit. In the paper on DEC's VAX [3], we described the importance of address size on architecture: "There is only one mistake that can be made in a computer design that is difficult to recover from—not providing enough address bits for memory addressing and memory management…" The 8086/8088 of the first IBM PCs had a 20-bit, or 1MB address space, the operating system using the remaining 384KB.

Concurrent with the introduction of the IBM PC, professional workstations were being created that used the Motorola 68000 CPU with its 32-bit data and address paths (4GB of maximum possible memory). Apple Computer used the Motorola "68K" in its Lisa and Macintosh machines. IBM's decision to use the Intel architecture with limited addressing, undoubtedly had the effect of impeding the PC by a decade as the industry waited for Intel to evolve architecture to support a larger address and virtual memory space. Hundreds of companies started up to build personal computers ("PC clones") based on the IBM PC reference design circa 1981. Dozens of companies also started to build workstations based on a 68K CPU running the UNIX operating system. This was the era of "JAWS" (Just Another WorkStation) to describe efforts at Apollo, HP, IBM, SGI, Sun Microsystems and others based on 32-bit versus 16-bit. Virtually all of these "workstations" were eliminated by simple economics as the PC—based on massive economies of scale and commoditization of both the operating system and all constituent hardware elements—evolved to have sufficient power and pixels.

"Minimal" CMOS Microsystems on a Chip circa 1990 Establish New Classes using Smaller, Less-Expensive, Chips. In 2007, many systems are composed of microprocessor components or "cores" with less than 50,000 transistors per microprocessor core at a time when the leading-edge microprocessor chips have a billion or more transistors (see Figure 3). Such cores using lower cost, less than the state-of-the-art chips and highly effective, rapid design tools allow new, minimal classes to emerge. PDAs, cameras, cell phones, and PADs have all been established using this minimal computer design style based on small cores. In 1990, the Advanced RISC Machine (ARM) formed from a collaboration between Acorn and Apple as the basis for embedded systems that are used as computing platforms and achieved two billion units per year in 2006. Other higher-volume microsystem platforms using 4-, 8-…64-bit architectures including MIPS exist as core architectures for building such systems as part of the very large embedded market.

Rapidly Evolving Killer CMOS Micros circa 1985 Overtake Bipolar ICs to Eliminate Established Classes. In the early 1980s, the phrase "killer micro" was introduced by members of the technical comput-

ing community as they saw how the more rapidly evolving CMOS micro would overtake bipolar-based minicomputers, mainframes, and supercomputers if they could be harnessed to operate as a single system and operate on a single program or workload.

In the *Innovator's Dilemma*, Christensen describes the death aspect basis of Bell's Law by contrasting two kinds of technologies [4]. Sustaining technology provides increasing performance, enabling improved products at the same price as previous models using slowly evolving technology; disruptive, rapidly evolving technology provides lower-priced products that are non-competitive with higher-priced sustaining class to create a unique market space. Over time, the performance of lesser-performing, faster-evolving products eventually overtakes the established, slowly evolving classes served by sustaining technology.

From the mid-1980s until 2000, over 40 companies were established and went out of business attempting to exploit the rapidly evolving CMOS microprocessors by interconnecting them in various ways. Cray, HP, IBM, SGI, and Sun Microsystems remain in 2008 to exploit massive parallelism through running a single program on a large number of computing nodes.

Two potentially disruptive technologies for new classes include:

- The evolving SFF devices such as cell phones are likely to have the greatest impact on personal computing, effectively creating a class. For perhaps most of the four billion non-PC users, a SFF device becomes their personal computer and communicator, wallet, or map, since the most common and often only use of PCs is for email and Web browsing—both stateless applications.
- The One Laptop Per Child project aimed at a $100 PC (actual cost $188 circa November 2007) is possibly disruptive as a "minimal" PC platform with just a factor-of-two cost reduction. This is achieved by substituting 1G of flash memory for rotating-disk-based storage, having a reduced screen size, a small main memory, and built-in mesh networking to reduce infrastructure cost, relying on the Internet for storage. An initial selling price of $188 for the OLPC XO-1 model—approximately half the price of the least-expensive PCs in 2008—is characteristic of a new sub-class. OLPC will be an interesting development since Microsoft's Vista requires almost an order of magnitude more system resources.

Future Challenges

The Challenge of Constant Price, 10–100 billion Transistors per Chip, for General-Purpose Computing. The future is not at all clear how such large, leading-edge chips will be used in general-purpose computers. The resilient and creative supercomputing and large-scale service center communities will exploit the largest multiple-core, multithreaded chips. There seems to be no upper bound these systems can utilize. However, without high-volume manufacturing, the virtuous cycle is stopped—in order to get the cost and benefit for clusters, a high-volume personal computer market must drive demand to reduce cost. In 2007, the degree of parallelism for personal computing in current desktop systems such as Linux and Vista is nil, which either indicates the impossibility of the task or the inadequacy of our creativity.

Several approaches for very large transistor count (approximately 10 billion transistor chips) could be:

- System with primary memory on a chip for reduced substantially lower-priced systems and greater demands;
- Graphics processing, currently handled by specialized chips, is perhaps the only well-defined application that is clearly able to exploit or absorb unlimited parallelism in a scalable fashion for the most expensive PCs (such as for gaming and graphical design);
- Multiple-core and multithreaded processor evolution for large systems;
- FPGAs that are programmed using inherently parallel hardware design languages like parallel C or Verilog that could provide universality that we have not previously seen; and
- Interconnected computers treated as software objects, requiring new application architectures.

Independent of how the chips are programmed, the biggest question is whether the high-volume PC market can exploit anything other than the first path in the preceding list. Consider the Carver Mead 11-year rule—the time from discovery and demonstration until use. Perhaps the introduction of a few transactional memory systems has started the clock using a programming methodology that claims to be more easily understood. A simpler methodology that can yield reliable designs by more programmers is essential in order to utilize these multiprocessor chips.

Will SFF Devices Impact Personal Computing? Users are likely to switch classes when the performance and functionality of a lesser-priced class is able to satisfy their needs and still increase functionality. Since the majority of PC use is for communication and Web access, evolving a SFF device as a single communicator for voice, email, and Web access is quite natural. Two things will happen to accelerate the development of the class: people who have never

used or are without PCs will use the smaller, simpler devices and avoid the PC's complexity; and existing PC users will adopt them for simplicity, mobility, and functionality. We clearly see these small personal devices with annual volumes of several hundred million units becoming the single universal device evolving from the phone, PDA, camera, personal audio/video device, Web browser, GPS and map, wallet, personal identification, and surrogate memory.

With every TV becoming a computer display, a coupled SFF becomes the personal computer for the remaining applications requiring large screens. Cable companies will also provide access via this channel as TV is delivered digitally.

Ubiquitous Wireless: WiFi, Cellular Services, and Wireless Sensor Nets. Unwiring the connection around the computer and peripherals, televisions, and other devices by high-speed radio links is useful but the function is "unwiring," and not platform creation. Near-Field Communication (NFC) using RF or magnetic coupling offers a new interface that can be used to communicate a person's identity that could form a new class for wallets and identity. However, most likely the communication channel and biometric technology taken together just increase the functionality of small devices.

Wireless Sensor Nets: New Platform, Network, and Applications. Combining the platform, wireless network, and interface into one to integrate with other systems by sensing and effecting is clearly a new class that has been forming since 2002 with a number of new companies that are offering unwiring, and hence reduced cost for existing applications, such as process, building, home automation, and control. Standards surrounding the 802.15.4 link that competes in the existing unlicensed RF bands with 802.11*xyz*, Bluetooth, and phone transmission are being established.

New applications will be needed for wireless sensor nets to become a true class versus just unwiring the world. If, for example, these chips become part of everything that needs to communicate in the whole IT hierarchy, a class will be established. They carry out three functions when part of a fixed environment or a moving object: sense/effect; recording of the state of a person or object (things such as scales, appliances, switches, thermometers, and thermostats) including its location and physical characteristics; and communication to the WiFi or other special infrastructure network for reporting. RFID is part of this potentially very large class of trillions. Just as billions of clients needed millions of servers, a trillion dust-sized wireless sensing devices will be coupled to a billion other computers.

Conclusion

Bell's Law explains the history of the computing industry based on the properties of computer classes and their determinants. This article has posited a general theory for the creation, evolution, and death of various priced-based computer classes that have come about through circuit and semiconductor technology evolution from 1951. The exponential transistor density increases forecast by Moore's Law [6] being the principle basis for the rise, dominance, and death of computer classes after the 1971 microprocessor introduction. Classes evolve along three paths: constant price and increasing performance of an established class; supercomputers—a race to build the largest computer of the day; and novel, lower-priced "minimal computers." A class can be subsumed by a more rapidly evolving, powerful, less-expensive class given an interface and functionality. In 2010, the powerful microprocessor will be the basis for nearly all classes from personal computers and servers costing a few thousand dollars to scalable servers costing a few hundred million dollars. Coming rapidly are billions of cell phones for personal computing and the tens of billions of wireless sensor networks to unwire and interconnect everything. As I stated at the outset, in the 1950s a person could walk inside a computer and by 2010 a computer cluster with millions of processors will have expanded to the size of a building. Perhaps more significantly, computers are beginning to "walk" inside of us. C

References

1. Bell, C.G. The mini and micro industries. *Computer 17*, 10 (Oct. 1984), 14–30.
2. Bell, C.G. Multis: A new class of multiprocessor computers. *Science 228* (Apr. 26, 1985) 462–467.
3. Bell, G. and Strecker, W. Computer structures: What have we learned from the PDP-11. *IEEE Computer Conference Proceedings* (Florida, Nov. 1975).
4. Christensen, C.M. *The Innovator's Dilemma.* Harvard Business School Press, 1997.
5. Gray, J. and Shenoy, P. Rules of thumb in data engineering. In *Proceedings of ICDE200* (San Diego, Mar. 1–4, 2000). IEEE press.
6. Moore, G.E. Cramming more components onto integrated circuits. *Electronics 8*, 39 (Apr. 19, 1965); revised 1975.
7. Nelson, D.L. and Bell, C.G. The evolution of workstations. *IEEE Circuits and Devices Magazine* (July 1986), 12–15.

Gordon Bell (gbell@microsoft.com) is a principal researcher in Microsoft Research Silicon Valley, working in the San Francisco Laboratory.

BY GRETA L. POLITES AND
RICHARD T. WATSON

THE CENTRALITY AND PRESTIGE OF CACM

Journal rankings identify the most respected publications in a field, and can influence which sources to read to remain current, as well as which journals to target when publishing. Ranking studies also help track the progress of the field, identifying core journals and research topics, and tracking changes in these topics and perceptions over time. Past journal ranking studies have consistently found *Communications of the ACM* (CACM) to be very highly respected within the IS discipline. However, the exact nature of its relationships to other IS journals has not been thoroughly examined. In this article, we report a social network analysis (SNA) of 120 journals for the purpose of exploring in detail CACM's position within the IS journal network.

SNA techniques "discover patterns of interaction between social actors in social networks" [6]. Common SNA procedures include information flow analysis (to determine the direction and strength of information flows through the network), calculation of centrality measures (to determine individual roles within a network), hierarchical clustering (to uncover cliques whose members are fully or almost fully connected), block modeling (to discover key links between different subgroups in a network), and calculation of structural equivalence measures (to identify network members with similar characteristics) [2, 6].

In the context of a citation network, SNA allows us to examine relationships between journals, identify roles played by individual journals, and identify cliques or subgroups of journals representing particular streams of research. Our study builds on prior SNA studies [see 1, 5] by presenting evidence of CACM's prominence within a relatively large network incorporating 120 journals that have appeared in previous IS journal ranking studies. CACM was found to be the top-ranked publication in the network based not only on the normalized number of citations received and information source criteria, but also on measures of local and global centrality.

Rank	Journal	Actual (Raw) # of Citations Received	Ranking Score Using Normalized Data	% of All Network Citations Received	# of All Network Journals Citing	% of All Network Journals Citing
1	Communications of the ACM	1961	9.904	0.085	101	0.849
2	Management Science	2226	6.506	0.056	71	0.597
3	MIS Quarterly	1694	4.811	0.041	55	0.462
4	Administrative Science Quarterly	1226	4.064	0.035	50	0.420
5	Harvard Business Review	1014	3.995	0.034	56	0.471
6	IEEE Transactions on Computers	639	3.641	0.031	60	0.504
7	IEEE Trans. on SW Engineering	650	3.441	0.030	52	0.437
8	Academy of Management Journal	1146	2.965	0.026	52	0.437
9	IEEE Transactions on PAM	734	2.888	0.025	26	0.218
10	Academy of Management Review	1009	2.870	0.025	49	0.412
11	IEEE Trans. on Info. Technology	1199	2.720	0.023	32	0.269
12	Artificial Intelligence	407	2.610	0.022	38	0.319
13	Organization Science	855	2.557	0.022	47	0.395
14	IEEE Computer	393	2.546	0.022	61	0.513
15	IEEE Trans. on Communications	1173	2.486	0.021	25	0.210
16	JASIS	373	2.415	0.021	29	0.244
17	Journal of the ACM	379	2.287	0.020	45	0.378
18	European J. of Operational Res.	633	2.117	0.018	38	0.319
19	Operations Research	837	2.074	0.018	33	0.277
20	IEEE Trans. on Sys., Man & Cyb.	480	2.064	0.018	46	0.387
21	Information Systems Research	587	1.848	0.016	45	0.378
22	Information Systems	519	1.791	0.015	46	0.387
23	Sloan Management Review	507	1.783	0.015	48	0.403
24	Information & Management	537	1.767	0.015	45	0.378
25	IEEE Transactions on KDE	259	1.627	0.014	46	0.387

Table 1. Normalized ranking scores for top 25 cited journals.

Methodology

Journal selection. Ideally, one would include all journals used by IS researchers when specifying the IS journal network. However, many IS journals are not currently indexed by ISI's Science Citation Index or Social Science Citation Index, and the resources required to collect manual data on these other journals is prohibitive. Thus for this study, we began with a subset of 125 journals listed on the ISWorld Journal Rankings Web page (which presents a composite ranking based on eight broad-based subjective and objective studies conducted between 1995–2005; see www.isworld.org/csaunders/rankings.htm). One significant departure we made from previous studies was to include IEEE and ACM Transactions and ACM SIG publications as individual entities within the network. This allowed us to track the actual citation patterns, contributions, and relationships associated with each individual publication. All ACM Transactions and SIG publications that were recorded in the 2003 *Journal Citation Reports* (*JCR*) were included in the study. IEEE Transactions appearing in *JCR* were filtered for their level of IS-specificity.

Only 91 of our initial 125 journals were indexed as "Cited Journals" in 2003. Focusing on the "Citing Journal" reports allowed us to capture data on an additional 36 journals (including individual IEEE/ACM Transactions and SIG publications) cited by the base 91, for a total of 127 journals. Citations made by the 36 non-indexed journals were tabulated manually, using either electronic or print copies of their 2003 articles. Journals that could not be located, or that had ceased publication prior to 2003, were removed from the list, leaving a final group of 120 journals.[1]

Many of the journals on the final list are not considered "IS-specific." However, a journal that is not a strong publication outlet for IS-related articles can still have a strong influence on the field. Thus the approach was to cast a wide net and allow SNA to identify which journals actually have a strong influence on various subcommunities or cliques of IS research.

[1]A complete list of these journals and their associated abbreviations is available from the authors on request.

Data standardization. After eliminating self-citations (including 157 self-citations for CACM, or 36% of its total citations within the network in 2003), we normalized the data to account for citation differences due to journals having different reference list criteria, longer articles, more frequent publishing cycles, or more articles per issue. Thus each cell in the normalized data matrix represents the proportion of a given journal's total network citations that were made to other journals in the network, with each "receiving" journal's final normalized score being the sum of these proportions [1, 3]. It is possible that some bias could be introduced by this method, particularly when citing journals make very few citations and a disproportionate percentage of these go to a single journal. Consistent with prior ranking studies, CACM overwhelmingly received the highest normalized score for any cited journal in the network (see Table 1).

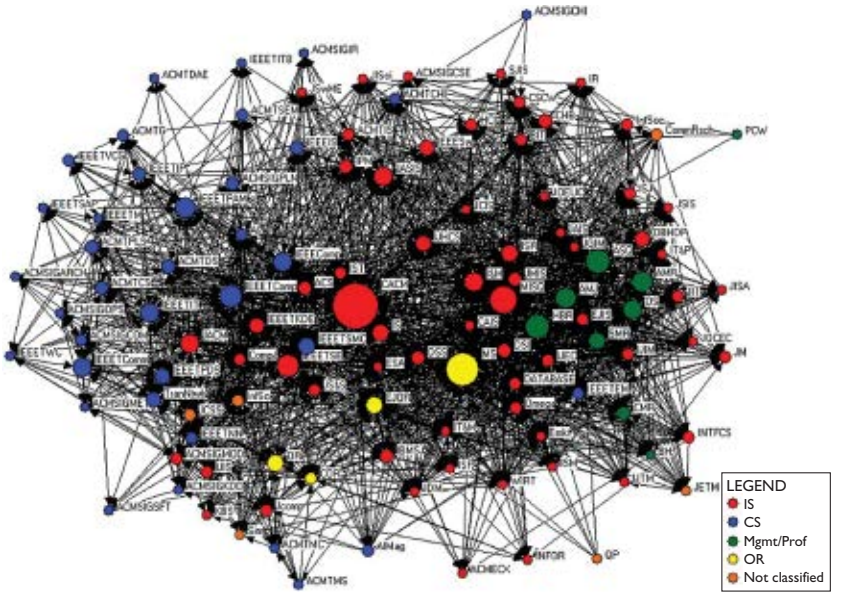


Figure 1. Information flow (full network).

Data Analysis

Information flow. Network citation data was analyzed using UCINET 6 and Netdraw 2.34. The journals were classified into the broad categories of IS, computer science, management/professional, and operations research based on classifications used in prior IS journal ranking studies.[2] Node size in the network diagrams is a function of each journal's normalized ranking, while arrow direction represents information flowing from cited journals to citing journals. The full network (see Figure 1) is split fairly evenly, with journals allied more closely to computer science (blue) on one side, and journals allied more closely with business disciplines, including IS (red), on the other. CACM occupies a central position in the full network, bridging the gap between the more technical computer science journals and the more business-oriented IS journals. *MIS Quarterly*, *Information Systems Research*, and *Journal of MIS*, which are usually considered the most prestigious "pure IS" journals, all occupy the dense central region of the red cluster.

We use spring embedding, a method that positions

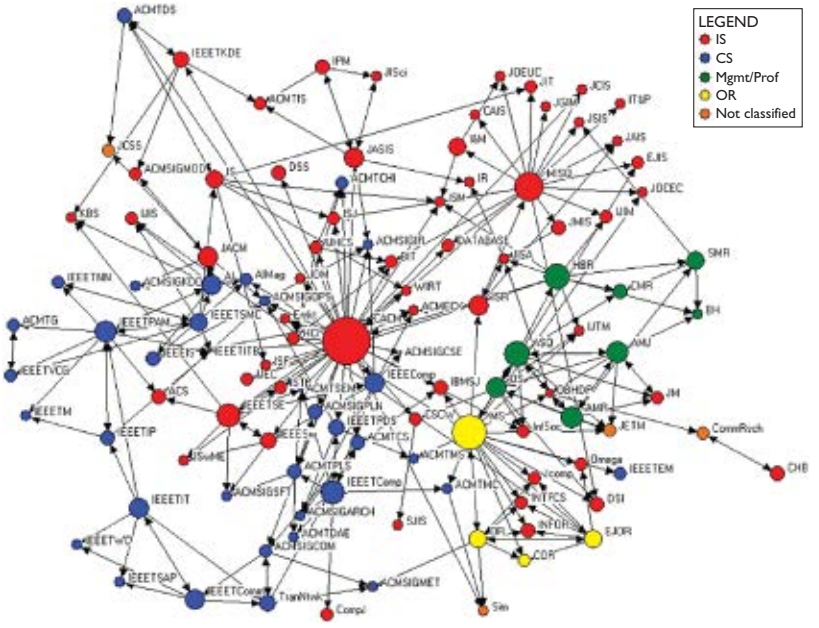


Figure 2. Information flow (spring embedding, 0.10 threshold).

network actors graphically based on their pairwise geodesic distances [1], to analyze the direction and strength of network information flows. At a threshold of 0.10 (meaning the proportion of Journal A's total citations that were made to Journal B is >= 0.10), clusters of related journals begin to emerge, as well as journals that serve as connections between clusters (see Figure 2). A "pure IS" cluster emerges (upper right-hand corner), which cites *MIS Quarterly* heavily but is

[2]Details on how we determined our classifications and the studies upon which these classifications are based are available on request.

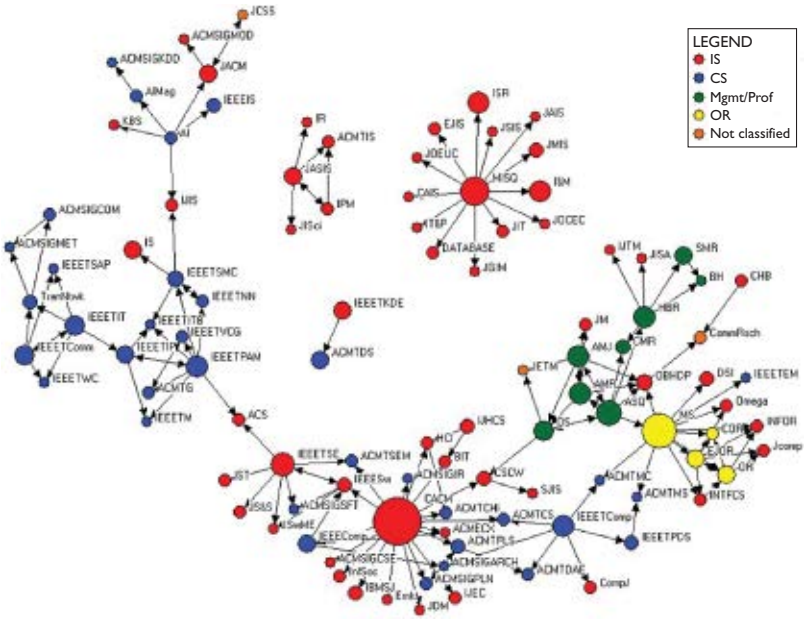


Figure 3. Information flow (spring embedding, 0.15 threshold).

largely isolated from other journals in the network. On the other hand, CACM continues to be a source of information to journals from both the computer science and IS camps, but is no longer an important source of information to the professional/managerial and operations research journal clusters.

Increasing the threshold from 0.10 to 0.15 does not greatly impact the view of CACM's relationships to other journals, although it does lead to a clearer overall delineation of network relationships (see Figure 3). By combining the information from these three flow diagrams, we infer that the journals giving the highest proportion of their overall citations to CACM in 2003 had the following research foci:

- Software engineering, data management, and computer systems;
- E-commerce;
- Collaboration and group support systems;
- Human-computer interaction;
- IT's role in society; and
- Emerging areas for IS research and development.

Information flow analysis can also be used to identify which network journals are the most important sources of information for CACM itself. Only two journals (*MIS Quarterly* and *Harvard Business Review*) received more than 10% of CACM's total citations within the network in 2003. Other journals receiving at least 4% of CACM's network citations include (in descending order) *IEEE Computer*, *Sloan Management Review*, *Management Science*, *Information Systems Review*, and *IEEE Software*, all of which are highly respected publications within their respective disciplines.

Information flow analysis also identifies information sources (journals that receive citations from more journals than they cite) and information sinks (journals that make citations to many different journals, but are not cited by as many in return). Overall, 52 journals were classified as information sources and 67 as information sinks. CACM received the second-highest information-source ranking in the network (net degree of 52, compared to a net degree of 55 for *Harvard Business Review*). Overall, CACM cited 49 journals in 2003 and was cited by 101. These scores are obviously highly sensitive to restrictions on reference lists or infrequent publication cycles. Some scores might also be affected by *JCR*'s practice of truncating most journal-to-journal citation counts less than two.

Prominence measures. An actor's prominence in a social network can be based on either centrality (visibility due to "extensive involvement in relations") or prestige (visibility based on "the extensive relations directed at them") [4]. In a journal citation network, it is more appropriate to assess prominence based on measures of prestige, rather than centrality.

Network actors can exhibit prominence due to having more direct ties with other actors (degree centrality/prestige), shorter path lengths to other actors (closeness centrality/prestige), or structurally advantageous positions between other actors (betweenness centrality/prestige) [2]. Degree is a localized measure providing information on an actor's immediate neighborhood, whereas closeness and betweenness are global measures.

Prominence measures are best interpreted in light of the overall level of network centralization. Centralization measures report a network's degree of centralization as a percentage of that of a perfectly centralized star network [2]. Thus the higher the network centralization, the more closely the network resembles a perfect star network with one central actor, indicating unequal distribution of power (or prestige) within the network. While the centralization of the IS journal network varies depending on the measure used, it tends to be moderate at best (ranging from 21.50% for betweenness measures to 66.45% for degree measures).

Freeman degree prestige is a common method of determining journal rankings, including our normal-

ized rankings in Table 1. Based on symmetric (reciprocated) citation patterns, CACM receives the highest-degree centrality ranking, counting 46 (38.66%) of 119 journals as being in its immediate neighborhood. Using asymmetric (in-directed) citation patterns, which are a measure of prestige rather than simple centrality, CACM again receives the highest ranking, with citations from 101 (84.87%) of the 119 journals, 30 greater than its nearest competitor.

A more insightful measure of degree prestige in a citation network is the Bonacich power index, which discriminates between citations received from more popular journals versus less popular journals (based on their respective degrees). The top 25 journals based on the Bonacich power index are shown in Table 2. There are several interesting differences between the Bonacich and Freeman degree results. Using the Freeman method, a large percentage of the top 25 journals have a management/professional orientation (including five of the top 10 journals). However, when using the Bonacich power method, many of these journals drop off the list, being replaced by more traditional IS journals. This implies that while management/professional journals are prestigious within their immediate neighborhood, their prestige does not carry over to the network as a whole. The Bonacich measure, however, is quite sensitive to the selected attenuation factor (0.6, in this case).

Bonacich Power (Beta = 0.06)			Information Centrality		
Rank	Journal	Raw Score	Rank	Journal	Normed Score
1	CACM	1193.010	1	CACM	0.678
2	DSS	972.685	2	MS	0.646
3	EJOR	943.964	3	MISQ	0.631
4	MS	922.228	4	HBR	0.613
5	IEEETSMC	848.698	5	ASQ	0.597
6	IEEETKDE	833.462	6	IEEETComp	0.589
7	I&M	817.880	6	IEEESE	0.589
8	IEEETSE	790.462	8	IEEEComp	0.578
9	MISQ	714.948	9	AMR	0.574
10	IEEETEM	709.908	10	OS	0.572
11	DSI	664.207	11	AMJ	0.567
12	Omega	637.369	12	IS	0.563
13	ACS	615.793	13	AI	0.562
14	COR	602.174	14	ISR	0.558
15	IJHCS	584.977	15	JACM	0.557
16	ISR	584.595	16	I&M	0.556
17	AI	558.125	17	SMR	0.554
18	JMIS	553.574	18	IEEETSMC	0.549
19	IS	501.022	19	EJOR	0.548
20	EJIS	492.562	19	IEEETPAM	0.548
21	IST	487.976	21	OR	0.539
22	IEEETComp	475.624	22	IJHCS	0.532
23	JS&S	463.307	23	IEEETKDE	0.531
24	IEEETPDS	462.803	24	JASIS	0.529
25	IEEEIS	440.927	25	DSI	0.524
---	-------	-------	25	IEEESw	0.524

Table 2. Most prestigious/central journals in the IS journal network.

CACM's top ranking, using the Bonacich power index, indicates the journals ranked close to it likewise have a high degree of prestige. It is also instructive to examine the types of journals not citing CACM in 2003; for the most part, these are academic and practitioner journals from the management discipline.

While it is possible to calculate closeness and betweenness centrality measures in citation networks, information centrality is considered a more appropriate measure to use in such cases, since information exchange between journals does not always follow the shortest path. Information centrality takes into account the actual strength of ties between actors, and thus indicates the relative drop in network efficiency brought about by removing a particular actor. Results for the top 25 journals appear in Table 2, and are highly correlated with the Freeman degree rankings. This makes sense, since in a citation network where the top journals tend to be widely cited, the removal of these journals will obviously cause a serious disruption to information flow. CACM is the most prominent journal in the network based on information centrality, once again highlighting its importance in dispersing knowledge throughout the network.

Limitations

This study only examined citations made by network journals in a single year (2003). However, Spearman rank correlation tests indicate statistically significant correlations between the normalized rankings in our single-year study and several past multiyear ranking studies, increasing our confidence in the results.[3] It is possible that the age of cited journals could impact the results, since older journals have more published articles available to be cited. In addition, the journal classifications (IS, CS, Mgmt/Prof, and OR) used here, while based on prior studies, have unclear boundaries. However, one of SNA's advantages is that it can in fact uncover subtle, unrecognized relationships between journals, and thus can aid in the development of more accurate classification schemes in the future.

Future Directions

The Internet has amplified the power of many existing networks and sustains a large collection of new networks (for example, the open source movement). Increasingly, there is recognition that network analy-

[3] A complete list of the 26 studies to which we compared our normalized rankings and the results of the corresponding correlation tests is available on request.

sis can tell us a great deal about the relationships between people and between entities. Search engines such as Google exploit the linkages within a network to determine the implicit ranking of pages. The same analytic method, as used in this article, enables us to reveal the importance of journals in the network of computing-related scholarship. Thus, this research provides guidance to scholars in assessing the importance of a particular outlet in a specific academic publication network.

In the age of the Internet, networks are dynamic and new technologies threaten the equilibrium of existing relationships. For example, journal reputations tend to be long-standing, but what is the effect of a potential reshaping of search strategies? In the days of paper-based journals, searching was often confined to those journals with the highest reputation. Now, facilities such as the ACM Portal and Google Scholar support searching across a wide range of journals. Electronic searching, compared to manual searching, means scholars can focus on finding highly relevant articles in a broad domain rather than restricting themselves to a small set of journals. As a result, we might see changes in journal relationships as a consequence of these new tools, and more broadly, we are likely to see changes in the nature of networks as a result of the adoption of new technologies for linking people and accessing information. C

References

1. Biehl, M., Kim, H., and Wade, M. Relationships among the academic business disciplines: A multi-method citation analysis. *Omega 34*, 4 (Aug. 2006), 359–371.
2. Hanneman, R.A. *Introduction to Social Network Methods*. University of California, Riverside, CA, 2001.
3. Holsapple, C.W. and Luo, W. A citation analysis of influences on collaborative computing research. *Computer Supported Cooperative Work 12*, 3 (Sept. 2003), 351–366.
4. Knoke, D. and Burt, R.S. Prominence. In Burt, R.S. and Minor, M.J., Eds., *Applied Network Analysis: A Methodological Introduction*. SAGE Publications, Beverly Hills, CA, 1983.
5. Nerur, S., Sikora, R., Mangalaraj, G., and Balijepally, V. Assessing the relative influence of journals in a citation network. *Commun. ACM 48*, 11 (Nov. 2005), 71–74.
6. Xu, J. and Chen, H. Criminal network analysis and visualization. *Commun. ACM 48*, 6 (June 2005), 101–107.

Greta L. Polites (gpolites@uga.edu) is a doctoral student in the MIS Department, Terry College of Business, at the University of Georgia, Athens.

Richard T. Watson (rwatson@terry.uga.edu) is the J. Rex Fuqua Distinguished Chair for Internet Strategy in the MIS Department, Terry College of Business, at the University of Georgia, Athens.

introducing...

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

Helping Members Meet Today's Career Challenges

Over 2,200 FREE Online Courses from SkillSoft

The ACM online course program features **free and unlimited access to over 2,200 online courses** from SkillSoft, a leading provider of e-learning solutions. This new collection of courses offers a host of valuable resources that will help to maximize your learning experience. Available on a wide range of information technology and business subjects, these courses are open to ACM Professional and Student Members.

SkillSoft courses offer a number of valuable features, including:

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

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

ACM Professional Member

600 FREE Online Books from Safari

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

500 FREE Online Books from Books24x7

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

Association for Computing Machinery

Advancing Computing as a Science & Profession

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

COMMUNICATIONS of the ACM

Communications of the ACM has a long history of attracting advertisers who value a global audience of computing professionals. Indeed, the first ad appeared by the fifth issue: The Radio Corporation of America (aka RCA) advertised on the back cover of the May 1958 issue to promote the many career opportunities available in its Astro-Electronics Products Division. That placement would draw the attention of many corporate competitors of the day, including Bendix, Remington Rand, Burroughs, IBM, Sylvania, and Wiley. All recognized the stellar audience their ads would reach if positioned in *Communications.*

Now, in its 50th year, advertisers continue to value the role *Communications* plays in keeping its readers ahead of the technological trends, new applications, and timely research developments. ACM's flagship publication is delivered each month to its 83,000 members—an illustrious audience representing every known computing discipline (and some still unknown) who look to their monthly edition to stay abreast of the latest in their own field as well as learn how advancements in other fields will impact their work. *Communications of the ACM* is devoted to providing readers with in-depth coverage of the uses, strengths, and future of information technology and computer science.

Competitive realities demand new skills from successful decision makers, IT managers, and researchers. New products enter the marketplace daily promising greater productivity and quality. But new products alone do not offer long-awaited profitability gains or offer a competitive edge—people do—well-informed people who regularly read *Communications of the ACM.*

Communications readers are highly educated, with 68% obtaining master's or doctorate degrees and 55% with more than 20 years of professional experience. According to a recent independent readership survey, the majority of *Communications* readers describe their primary job responsibilities as software/applications designers, developers, and engineers. Indeed, 75% of *Communications* readers design or write software. The average annual budget for computer products at the companies these readers represent is over $850,000.

As *Communications of the ACM* embarks on a new year, with many new and extraordinary editorial changes afoot, its readers will still turn to it for the same reasons those first readers did 50 years ago—to stay informed of the ever-changing world of computing and the countless opportunities it affords.

"Communications readers describe their primary job responsibilities as software/applications designers, developers, and engineers. 75% of Communications readers design or write software"

Association for Computing Machinery

Advancing Computing as a Science & Profession

2008 Display Advertising Rates Communications of the ACM

Size/Frequency	1X	3X	6X	9X	12X	18X	24X
BLACK/WHITE- FULL PAGE	8500	8300	8100	7900	7700	7500	7300
2/3 PAGE	7100	7000	6800	6600	6400	6200	6000
1/2 PAGE ISLAND	6300	6200	6100	5900	5700	5200	5000
1/2 PAGE	6000	5800	5700	5500	5300	5100	4800
1/3 PAGE	4900	4880	4700	4500	4300	4100	3900
1/4 PAGE	3800	3750	3700	3650	3600	3400	3350
1/6 PAGE	2800	2750	2700	2650	2550	2500	2450
4-COLOR-FULL PAGE	10000	9500	9400	9300	9100	8800	8600
2/3 PAGE	8400	8300	8100	7900	7750	7500	7250
1/2 PAGE ISLAND	7600	7500	7400	7200	7000	6500	6300
1/2 PAGE	7300	7100	7000	6800	6600	6400	6200
1/3 PAGE	6240	6200	6000	5800	5600	5400	5200
1/4 PAGE	5100	5000	4900	4850	4800	4700	4650
1/6 PAGE	4100	4000	3900	3850	3800	3750	3700
COVER 2&3- FULL PAGE	10500	10450	10400	10350	10300	10250	10200
COVER 4- FULL PAGE	11000	10700	10600	10500	10400	10200	10000

2008 Recruitment Advertising Rates	1X	3X	6X	9X	12X	18X	24X
BLACK/WHITE- FULL PAGE	8200	8000	7800	7600	7400	7200	7000
2/3 PAGE	6800	6650	6500	6350	6200	6050	5600
1/2 PAGE ISLAND	6000	5900	5750	5500	5250	5000	4750
1/2 PAGE	5800	5650	5500	5300	5000	4800	4600
1/3 PAGE	4600	4500	4450	4400	4300	4100	3700
1/4 PAGE	3700	3600	3500	3450	3400	3350	3200
1/6 PAGE	2700	2600	2500	2450	2400	2350	2200
4-COLOR-FULL PAGE	9500	9300	9100	8800	8600	8500	8300
2/3 PAGE	8100	8000	7800	7600	7400	7200	7000
1/2 PAGE ISLAND	7400	7300	7100	6900	6800	6300	6100
1/2 PAGE	7000	6900	6800	6600	6300	6200	5900
1/3 PAGE	6000	5900	5800	5600	5500	5300	5000
1/4 PAGE	5000	4900	4800	4700	4600	4500	4400
1/6 PAGE	4000	3900	3800	3750	3700	3600	3550

Ad sizes and formats

Trim Size	8 1/8”x 10 7/8”
Sizes (live area)	
Full PAGE	7”x10”
2/3 PAGE	4 5/16” x 10”
1/2 PAGE	7” x 4 5/8”
1/2 PAGE ISLAND	3 7/16”x 10”
1/3 PAGE	4 5/8” x 4 3/4”
1/4 PAGE	3 7/16” x 4 3/4”
1/6 PAGE	2 1/4” x 4 7/8”

FORMAT:
All display advertising must be provided in a digital format. Required format for both black-and-white and color advertisements is a high-resolution Adobe Acrobat PDF file format (version 6 and compatible) with all fonts embedded.

The reproduction quality of color advertisements printed in *Communications* will depend on the quality of the supplied digital files and proofs.

Publication Closing Dates:

Issue	**Deadline**
January 08	Nov 20, 2007
February 08	Dec 19, 2007
March 08	Jan 18, 2008
April 08	Feb 20, 2008
May 08	Mar 19, 2008
June 08	Apr 18, 2008
July 08	May 19, 2008
August 08	June 20, 2008
September 08	July 18, 2008
October 08	Aug 20, 2008
November 08	Sept 19, 2008
December 08	Oct 20, 2008

SUBMIT ADS:
ACM Advertising Department
William R. Kooney, Account Executive
Email: kooney@acm.org
212-626-0687 v
212-869-0481 f

A preview of things to come.

BREAKTHROUGH RESEARCH

Cutting-edge computing research, as this issue attests, is at the very foundation of CACM's evolution. Over the years, the presentation of computing research within this publication has taken different forms, and will do so once again.

A new CACM editorial model, as detailed in Moshe Vardi's article in this issue (see page 44), calls for research to make a very real comeback, but with a twist. The research articles presented in future issues of CACM will maintain their detailed integrity, but be written for a broad audience. Moreover, each article will be preceded by a single-page Technical Perspective that notes the significance of the research and the impact that research may have on the computing field.

Here we present two examples of the kind of research articles and accompanying Technical Perspectives planned for CACM later this year.

Technical Perspective:

The Data Center Is The Computer

by David A. Patterson

Internet services are already significant forces in searching, retail purchases, music downloads, and auctions. One vision of 21st century IT is that most users will be accessing such services over a descendant of the cell phone rather than running shrink-wrapped software on a descendant of the PC.

There are dramatic differences between of developing software for millions to use as a service versus distributing software for millions to run their PCs. First, services must be always available, so dependability is critical. Second, services must have tremendous bandwidth to support many users, but they must also have low latency so as not annoy customers who can easily switch to competing services. Third, the companies can innovate more quickly because their software is only run inside the company.

These requirements have led to distributed data centers. They are distributed to prevent a site disaster resulting in loss of power or networking from stopping the service, and to reduce latency to a worldwide customer base.

Companies like Google are starting to hire computer architects. When I asked Luiz Barroso of Google why, he said, "The data center is now the computer." Hence, computer architects are now designing and evaluating data centers.

This is certainly a provocative notion. However, if the data center is the computer, it leads to the even more intriguing question "What is the equivalent of the ADD instruction for a data center?"

Jeffrey Dean and Sanjay Ghemawat offer one answer in their paper. Like the early days of computing in the mid 20th century, the early developers of services at Google had to worry about a myriad of gritty details. For Google it includes partitioning data sets, communicating between independent computers, scheduling tasks to run simultaneously, and handling hardware and software failures. Given the pain of that experience, and the desire to let many develop new services inside Google, Dean and Ghemawat chose to raise the level of abstraction. Like their 20th century predecessors, the art was in hiding minutia while still delivering good performance and a useful programming interface.

This brings to mind three questions:

- What are useful programming abstractions for such a large system?
- How do thousands of computers behave differently from a small system?
- What must you do differently to run the abstraction on thousands of computers?

They decided to offer a two-phase primitive. The first phase maps a user supplied function onto thousands of computers. The second phase reduces the returned values from all those thousands of instances into a single result. Note that these two phases are highly parallel yet simple to understand. Borrowing the name from a similar function in Lisp, they christened the primitive "MapReduce.

Heterogeneity is one difference between running MapReduce on a single computer versus thousands. Companies don't buy thousands of computers in one fell swoop, so a single data center will have generations of computers of varying speed processors and different amounts of DRAM and disk. In addition to hardware variety, even identical equipment will behave differently. Some of it will break before and perhaps while running your program. There will also be several computers that are limping along, making much slower progress than their siblings do.

What should MapReduce do in light of this challenging environment? Here are a few highlights. First, the scheduler accommodates dynamic variation by assigning tasks based on how well a computer has done recently rather than by a static priority. Second, to cope with laggard computations, it was much faster to re-execute them on the fast nodes than to wait for the slow ones to complete. Third, the Google File System allows programs to access files efficiently from any computer, so functions can be mapped everywhere.

To test performance, they ran a program across a data center that sorts 10 billion randomly-generated records in 10 to 15 minutes, despite node failures. That's an impressive 10 million records a second. Such performance allowed Google to replace the old ad hoc programs that regenerate Google's index of the Internet with faster and simpler code based on MapReduce.

In addition to production use of MapReduce, it empowered novice programmers. In a half hour they can now write, run, and see output from their programs run on thousands of computers. The original paper had just a few months of experience by novice users, so I was delighted to read the updated paper to learn what happened over the last two years.

The beauty of MapReduce is that any programmer can understand it, and its power comes from being able to harness thousands of computers behind that simple interface. When paired with the distributed Google File System to deliver data, programmers can write simple functions that can do amazing things.

I predict MapReduce will inspire new ways of thinking about the design and programming of large distributed systems. If MapReduce is the first instruction of the "data center computer," I can't wait to see the rest of the instruction set, as well as the data center programming language, the data center operating system, the data center storage systems, and more.

Biography

David Patterson (pattrsn@eecs.berkeley.edu) is the Pardee Professor of Computer Science at U. C. Berkeley, and is a Fellow and Past President of ACM.

World Class Journals from ACM

ACM Transactions on Autonomous and Adaptive Systems

http://taas.acm.org/

TAAS **addresses foundational, engineering, and technological aspects of complex computing systems exhibiting autonomous and adaptive behavior; including the understanding, development, and control of such systems.**

Order Codes: Print – 158 Online – 258
ISSN: 1556-4665
Pricing: $ 40 Professional
$ 35 Student
$ 140 Non-Member
$ 16 Air Service

ACM Transactions on The Web

http://tweb.acm.org/

TWEB **publishes research on web content, applications, use, and related enabling technologies; such as browsers and web interfaces; electronic commerce; electronic publishing; etc.**

Order Codes: Print – 159 Online – 259
ISSN: 1559-1131
Pricing: $ 40 Professional
$ 35 Student
$ 140 Non-Member
$ 16 Air Service

ACM Transactions on Knowledge Discovery from Data

http://tkdd.acm.org/

TKDD **publishes papers on a research in the knowledge discovery and analysis of diverse forms of data; such as scalable and effective algorithms for data mining and data warehousing, etc.**

Order Codes: Print – 170 Online – 270
ISSN: 1556-4681
Pricing: $ 50 Professional
$ 45 Student
$ 150 Non-Member
$ 16 Air Service

*Air Service is for residents outside North America only.

ACM publishes over 40 magazines and journals that cover a vast array of established as well as emerging areas of the computing field. IT professionals worldwide depend on ACM's publications to keep them abreast of the latest technological developments and industry news in a timely, comprehensive manner of the highest quality and integrity. For a complete listing of ACM's leading magazines & journals, including our renowned *Transaction Series*, please visit the ACM publications homepage at:

www.acm.org/pubs

PLEASE CONTACT ACM MEMBER SERVICES TO PLACE AN ORDER

Phone: 1.800.342.6626 (U.S. and Canada)
+1.212.626.0500 (Global)
Fax: +1.212.944.1318
(Hours: 8:30 AM – 4:30 PM, Eastern Time)
Email: acmhelp@acm.org
Mail: ACM Member Services
General Post Office
PO Box 30777
New York, NY 10087-0777 USA

Association for Computing Machinery

Advancing Computing as a Science & Profession

MapReduce: Simplified Data Processing on Large Clusters

by Jeffrey Dean and Sanjay Ghemawat

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a *map* and a *reduce* function, and the underlying runtime system automatically parallelizes the computation across large-scale clusters of machines, handles machine failures, and schedules inter-machine communication to make efficient use of the network and disks. Programmers find the system easy to use: more than ten thousand distinct MapReduce programs have been implemented internally at Google over the past four years, and an average of one hundred thousand MapReduce jobs are executed on Google's clusters every day, processing a total of more than twenty petabytes of data per day.

1 Introduction

Prior to our development of MapReduce, the authors and many others at Google implemented hundreds of special-purpose computations that process large amounts of raw data, such as crawled documents, Web request logs, etc., to compute various kinds of derived data, such as inverted indices, various representations of the graph structure of Web documents, summaries of the number of pages crawled per host, and the set of most frequent queries in a given day. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a map operation to each logical record' in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately. Our use of a functional model with user-specified map and reduce operations allows us to parallelize large computations easily and to use reexecution as the primary mechanism for fault tolerance.

Biographies

Jeff Dean (jeff@google.com) is a Google Fellow and is currently working on a large variety of large-scale distributed systems at Google's Mountain View, CA, facility.

Sanjay Ghemawat (sanjay@google.com) is a Google Fellow and works on the distributed computing infrastructure used by most the company's products. He is based at Google's Mountain View, CA, facility.

The major contributions of this work are a simple and powerful interface that enables automatic parallelization and distribution of large-scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs. The programming model can also be used to parallelize computations across multiple cores of the same machine.

Section 2 describes the basic programming model and gives several examples. In Section 3, we describe an implementation of the MapReduce interface tailored towards our cluster-based computing environment. Section 4 describes several refinements of the programming model that we have found useful. Section 5 has performance measurements of our implementation for a variety of tasks. In Section 6, we explore the use of MapReduce within Google including our experiences in using it as the basis for a rewrite of our production indexing system. Section 7 discusses related and future work.

2 Programming Model

The computation takes a set of *input* key/value pairs, and produces a set of *output* key/value pairs. The user of the MapReduce library expresses the computation as two functions: map and reduce.

Map, written by the user, takes an input pair and produces a set of *intermediate* key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key *I* and passes them to the reduce function.

The reduce function, also written by the user, accepts an intermediate key *I* and a set of values for that key. It merges these values together to form a possibly smaller set of values. Typically just zero or one output value is produced per reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory.

2.1 Example

Consider the problem of counting the number of occurrences of each word in a large collection of documents. The user would write code similar to the following pseudocode.

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");

reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));
```

The `map` function emits each word plus an associated count of occurrences (just 1 in this simple example). The `reduce` function sums together all counts emitted for a particular word.

In addition, the user writes code to fill in a *mapreduce specification* object with the names of the input and output files and optional tuning parameters. The user then invokes the *MapReduce* function, passing it to the specification object. The user's code is linked together with the MapReduce library (implemented in C++). Our original MapReduce paper contains the full program text for this example [8].

More than ten thousand distinct programs have been implemented using MapReduce at Google, including algorithms for large-scale graph processing, text processing, data mining, machine learning, statistical machine translation, and many other areas. More discussion of specific applications of MapReduce can be found elsewhere [8, 16, 7].

2.2 Types

Even though the previous pseudocode is written in terms of string inputs and outputs, conceptually the map and reduce functions supplied by the user have associated types.

map	(k1,v1)	→ list(k2,v2)
reduce	(k2,list(v2))	→ list(v2)

That is, the input keys and values are drawn from a different domain than the output keys and values. Furthermore, the intermediate keys and values are from the same domain as the output keys and values.

3. Implementation

Many different implementations of the MapReduce interface are possible. The right choice depends on the environment. For example, one implementation may be suitable for a small shared-memory machine, another for a large NUMA multiprocessor, and yet another for an even larger collection of networked machines. Since our original article, several open source implementations of MapReduce have been developed [1, 2], and the applicability of MapReduce to a variety of problem domains has been studied [7, 16].

This section describes our implementation of MapReduce that is targeted to the computing environment in wide use at Google: large clusters of commodity PCs connected together with switched Gigabit Ethernet [4]. In our environment, machines are typically dual-processor x86 processors running Linux, with 4-8GB of memory per machine. Individual machines typically have 1 gigabit/second of network bandwidth, but the overall bisection bandwidth available per machine is considerably less than 1 gigabit/second. A computing cluster contains many thousands of machines, and therefore machine failures are common. Storage is provided by inexpensive IDE disks attached directly to individual machines. GFS, a distributed file system developed in-house [10], is used to manage the data stored on these disks. The file system uses replication to provide availability and reliability on top of unreliable hardware.

Users submit jobs to a scheduling system. Each job consists of a set of tasks, and is mapped by the scheduler to a set of available machines within a cluster.

3.1 Execution Overview

The map invocations are distributed across multiple machines by automatically partitioning the input data into a set of M *splits*. The input splits can be processed in parallel by different machines. Reduce invocations are distributed by partitioning the intermediate key space into R pieces using a partitioning function (e.g., *hash*(*key*) **mod** R). The number of partitions (R) and the partitioning function are specified by the user.

Figure 1 shows the overall flow of a MapReduce operation in our implementation. When the user program calls the `MapReduce` function, the following sequence of actions occurs (the numbered labels in Figure 1 correspond to the numbers in the following list).

1. The MapReduce library in the user program first splits the input files into M pieces of typically 16-64MB per piece (controllable by the user via an optional parameter). It then starts up many copies of the program on a cluster of machines.

2. One of the copies of the program—the master— is special. The rest are workers that are assigned work by the master. There are M map tasks and R reduce tasks to assign. The master picks idle workers and assigns each one a map task or a reduce task.

3. A worker who is assigned a map task reads the contents of the corresponding input split. It parses key/value pairs out of the input data and passes each pair to the user-defined map function. The intermediate key/value pairs produced by the map function are buffered in memory.

4. Periodically, the buffered pairs are written to local disk, partitioned into R regions by the partitioning function. The locations of these buffered pairs on the local disk are passed back to the master who is responsible for forwarding these locations to the reduce workers.

5. When a reduce worker is notified by the master about these locations, it uses remote procedure calls to read the buffered data from the local disks of the map workers. When a reduce worker has read all intermediate data for its partition, it sorts it by the intermediate keys so that all occurrences of the same key are grouped together. The sorting is needed because typically many different keys map to the same reduce task. If the amount of intermediate data is too large to fit in memory, an external sort is used.

6. The reduce worker iterates over the sorted intermediate data and for each unique intermediate key encountered, it passes the key and the corresponding set of intermediate values to the user's reduce function. The output of the reduce function is appended to a final output file for this reduce partition.

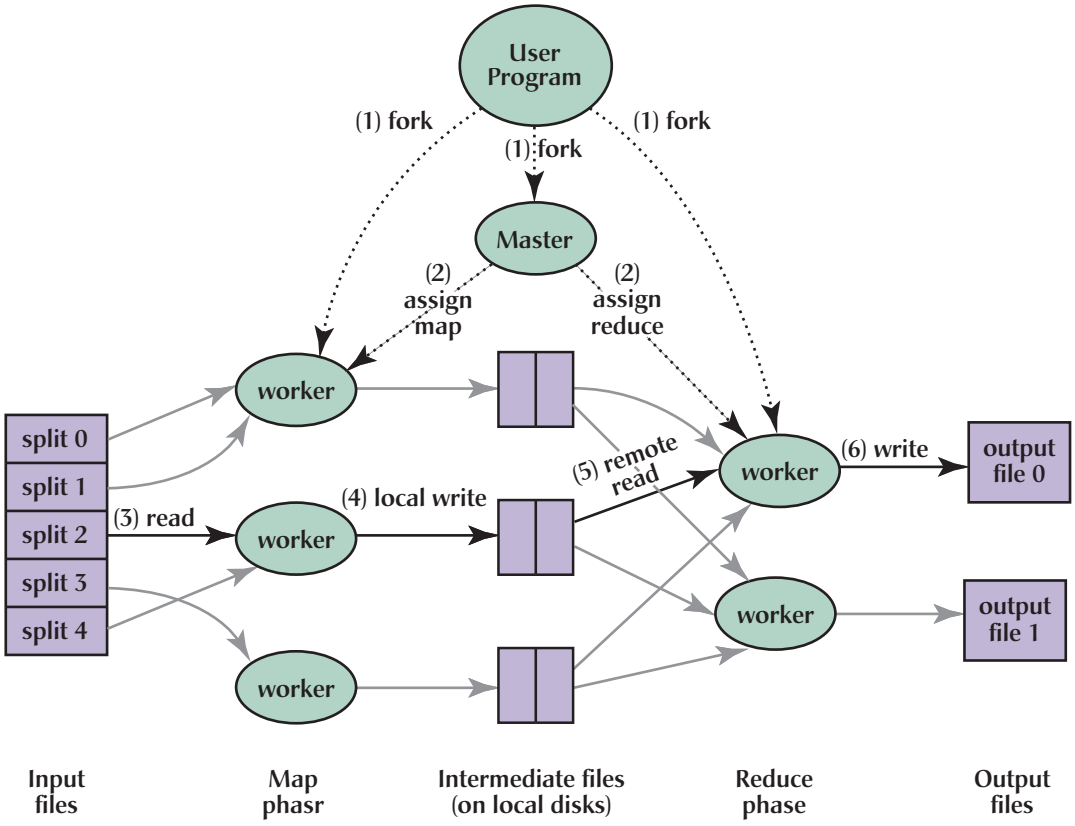


Fig. 1. Execution overview.

7. When all map tasks and reduce tasks have been completed, the master wakes up the user program. At this point, the `MapReduce` call in the user program returns back to the user code.

After successful completion, the output of the mapreduce execution is available in the *R* output files (one per reduce task, with file names specified by the user). Typically, users do not need to combine these *R* output files into one file; they often pass these files as input to another MapReduce call or use them from another distributed application that is able to deal with input that is partitioned into multiple files.

3.2 Master Data Structures

The master keeps several data structures. For each map task and reduce task, it stores the state (*idle*, *in-progress*, or *completed*) and the identity of the worker machine (for nonidle tasks).

The master is the conduit through which the location of intermediate file regions is propagated from map tasks to reduce tasks. Therefore, for each completed map task, the master stores the locations and sizes of the *R* intermediate file regions produced by the map task. Updates to this location and size information are received as map tasks are completed. The information is pushed incrementally to workers that have *in-progress* reduce tasks.

3.3 Fault Tolerance

Since the MapReduce library is designed to help process very large amounts of data using hundreds or thousands of machines, the library must tolerate machine failures gracefully.

Handling Worker Failures

The master pings every worker periodically. If no response is received from a worker in a certain amount of time, the master marks the worker as failed. Any map tasks completed by the worker are reset back to their initial *idle* state and therefore become eligible for scheduling on other workers. Similarly, any map task or reduce task in progress on a failed worker is also reset to *idle* and becomes eligible for rescheduling.

Completed map tasks are reexecuted on a failure because their output is stored on the local disk(s) of the failed machine and is therefore inaccessible. Completed reduce tasks do not need to be reexecuted since their output is stored in a global file system.

When a map task is executed first by worker *A* and then later executed by worker *B* (because *A* failed), all workers executing reduce tasks are notified of the reexecution. Any reduce task that has not already read the data from worker *A* will read the data from worker *B*.

MapReduce is resilient to large-scale worker failures. For example, during one MapReduce operation, network maintenance on a running cluster was causing groups of 80 machines at a time to become unreachable for several minutes. The MapReduce master simply reexecuted the work done by the unreachable worker machines and continued to make forward progress, eventually completing the MapReduce operation.

Semantics in the Presence of Failures

When the user-supplied map and reduce operators are deterministic functions of their input values, our distributed implementation produces the same output as would have been produced by a nonfaulting sequential execution of the entire program.

We rely on atomic commits of map and reduce task outputs to achieve this property. Each in-progress task writes its output to private temporary files. A reduce task produces one such file, and a map task produces R such files (one per reduce task). When a map task completes, the worker sends a message to the master and includes the names of the R temporary files in the message. If the master receives a completion message for an already completed map task, it ignores the message. Otherwise, it records the names of R files in a master data structure.

When a reduce task completes, the reduce worker atomically renames its temporary output file to the final output file. If the same reduce task is executed on multiple machines, multiple rename calls will be executed for the same final output file. We rely on the atomic rename operation provided by the underlying file system to guarantee that the final file system state contains only the data produced by one execution of the reduce task.

The vast majority of our map and reduce operators are deterministic, and the fact that our semantics are equivalent to a sequential execution in this case makes it very easy for programmers to reason about their program's behavior. When the map and/or reduce operators are nondeterministic, we provide weaker but still reasonable semantics. In the presence of nondeterministic operators, the output of a particular reduce task R_1 is equivalent to the output for R_1 produced by a sequential execution of the nondeterministic program. However, the output for a different reduce task R_2 may correspond to the output for R_2 produced by a different sequential execution of the nondeterministic program.

Consider map task M and reduce tasks R_1 and R_2. Let $e(R_i)$ be the execution of R_1 that committed (there is exactly one such execution). The weaker semantics arise because $e(R_1)$ may have read the output produced by one execution of M, and $e(R_2)$ may have read the output produced by a different execution of M.

3.4 Locality

Network bandwidth is a relatively scarce resource in our computing environment. We conserve network bandwidth by taking advantage of the fact that the input data (managed by GFS [10]) is stored on the local disks of the machines that make up our cluster. GFS divides each file into 64MB blocks and stores several copies of each block (typically 3 copies) on different machines. The MapReduce master takes the location information of the input files into account and attempts to schedule a map task on a machine that contains a replica of the corresponding input data. Failing that, it attempts to schedule a map task near a replica of that task's input data (e.g., on a worker machine that is on the same network switch as the machine containing the data). When running large MapReduce operations on a significant fraction of the workers in a cluster, most input data is read locally and consumes no network bandwidth.

3.5 Task Granularity

We subdivide the map phase into M pieces and the reduce phase into R pieces as described previously. Ideally, M and R should be much larger than the number of worker machines. Having each worker perform many different tasks improves dynamic load balancing and also speeds up recovery when a worker fails: the many map tasks it has completed can be spread out across all the other worker machines.

There are practical bounds on how large M and R can be in our implementation since the master must make $O(M+R)$ scheduling decisions and keep $O(M*R)$ state in memory as described. (The constant factors for memory usage are small, however. The $O(M*R)$ piece of the state consists of approximately one byte of data per map task/reduce task pair.)

Furthermore, R is often constrained by users because the output of each reduce task ends up in a separate output file. In practice, we tend to choose M so that each individual task is roughly 16MB to 64MB of input data (so that the locality optimization described previously is most effective), and we make R a small multiple of the number of worker machines we expect to use. We often perform MapReduce computations with M=200,000 and R=5,000, using 2,000 worker machines.

3.6 Backup Tasks

One of the common causes that lengthens the total time taken for a MapReduce operation is a straggler, that is, a machine that takes an unusually long time to complete one of the last few map or reduce tasks in the computation. Stragglers can arise for a whole host of reasons. For example, a machine with a bad disk may experience frequent correctable errors that slow its read performance from 30MB/s to 1MB/s. The cluster scheduling system may have scheduled other tasks on the machine, causing it to execute the MapReduce code more slowly due to competition for CPU, memory, local disk, or network bandwidth. A recent problem we experienced was a bug in machine initialization code that caused processor caches to be disabled: computations on affected machines slowed down by over a factor of one hundred.

We have a general mechanism to alleviate the problem of stragglers. When a MapReduce operation is close to completion, the master schedules backup executions of the remaining *in-progress* tasks. The task is marked as completed whenever either the primary or the backup execution completes. We have tuned this mechanism so that it typically increases the computational resources used by the operation by no more than a few percent. We have found that this significantly reduces the time to complete large MapReduce operations. As an example, the sort program described in Section 5.3 takes 44% longer to complete when the backup task mechanism is disabled.

4 Refinements

Although the basic functionality provided by simply writing map and reduce functions is sufficient for most needs, we have found a few extensions useful. These include:

- user-specified partitioning functions for determining the mapping of intermediate key values to the R reduce shards;
- ordering guarantees: Our implementation guarantees that within each of the R reduce partitions, the intermediate key/value pairs are processed in increasing key order;
- user-specified combiner functions for doing partial combination of generated intermediate values with the same key within the same map task (to reduce the amount of intermediate data that must be transferred across the network);
- custom input and output types, for reading new input formats and producing new output formats;
- a mode for execution on a single machine for simplifying debugging and small-scale testing.

The original article has more detailed discussions of each of these items [8].

5 Performance

In this section, we measure the performance of MapReduce on two computations running on a large cluster of machines. One computation searches through approximately one terabyte of data looking for a particular pattern. The other computation sorts approximately one terabyte of data.

These two programs are representative of a large subset of the real programs written by users of MapReduce—one class of programs shuffles data from one representation to another, and another class extracts a small amount of interesting data from a large dataset.

5.1 Cluster Configuration

All of the programs were executed on a cluster that consisted of approximately 1800 machines. Each machine had two 2GHz Intel Xeon processors with Hyper-Threading enabled, 4GB of memory, two 160GB IDE disks, and a gigabit Ethernet link. The machines were arranged in a two-level tree-shaped switched network with approximately 100-200Gbps of aggregate bandwidth available at the root. All of the machines were in the same hosting facility and therefore the round-trip time between any pair of machines was less than a millisecond.

Out of the 4GB of memory, approximately 1-1.5GB was reserved by other tasks running on the cluster. The programs were executed on a weekend afternoon when the CPUs, disks, and network were mostly idle.

5.2 Grep

The *grep* program scans through 10^{10} 100-byte records, searching for a relatively rare three-character pattern (the pattern occurs in 92,337 records). The input is split into approximately 64MB pieces ($M = 15000$), and the entire output is placed in one file ($R = 1$).

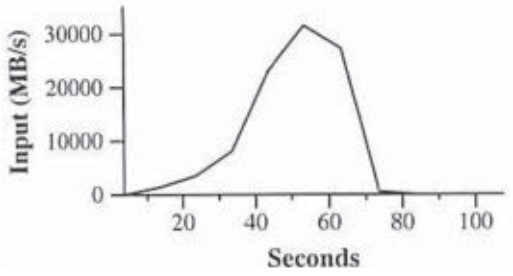


Fig. 2. Data transfer rate over time (mr-grep).

Figure 2 shows the progress of the computation over time. The Y-axis shows the rate at which the input data is scanned. The rate gradually picks up as more machines are assigned to this MapReduce computation and peaks at over 30 GB/s when 1764 workers have been assigned. As the map tasks finish, the rate starts dropping and hits zero about 80 seconds into the computation. The entire computation takes approximately 150 seconds from start to finish. This includes about a minute of startup overhead. The overhead is due to the propagation of the program to all worker machines and delays interacting with GFS to open the set of 1000 input files and to get the information needed for the locality optimization.

5.3 Sort

The sort program sorts 10^{10} 100-byte records (approximately 1 terabyte of data). This program is modeled after the TeraSort benchmark [12].

The sorting program consists of less than 50 lines of user code. The final sorted output is written to a set of 2-way replicated GFS files (i.e., 2 terabytes are written as the output of the program).

As before, the input data is split into 64MB pieces ($M = 15000$). We partition the sorted output into 4000 files ($R = 4000$). The partitioning function uses the initial bytes of the key to segregate it into one of pieces.

Our partitioning function for this benchmark has built-in knowledge of the distribution of keys. In a general sorting program, we would add a prepass MapReduce operation that would collect a sample of the keys and use the distribution of the sampled keys to compute split-points for the final sorting pass.

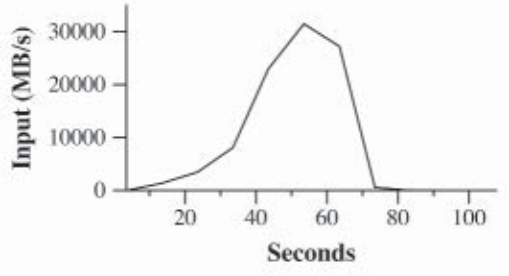


Figure 2: Data transfer rate over time (mr-grep)

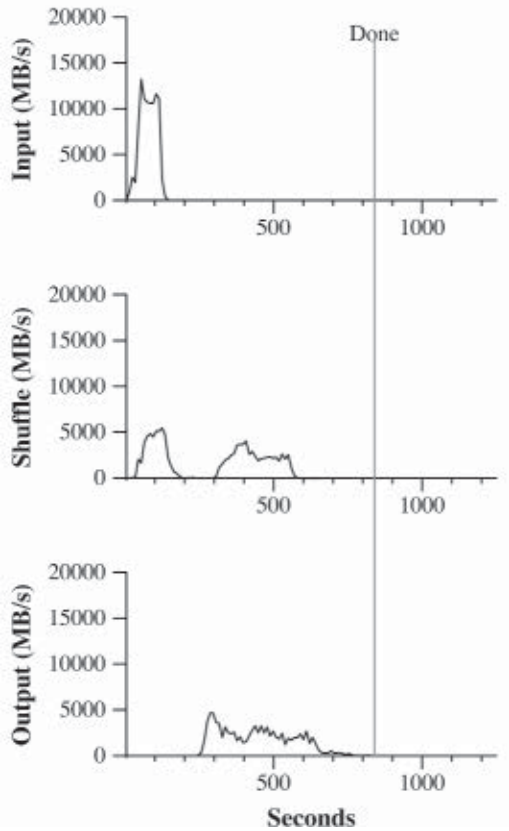


Fig. 3. Data transfer rate over time (mr-sort).

Figure 3 shows the progress of a normal execution of the sort program. The top-left graph shows the rate at which input is read. The rate peaks at about 13GB/s and dies off fairly quickly since all map tasks finish before 200 seconds have elapsed. Note that the input rate is less

than for *grep*. This is because the sort map tasks spend about half their time and I/O bandwidth writing intermediate output to their local disks. The corresponding intermediate output for grep had negligible size.

A few things to note: the input rate is higher than the shuffle rate and the output rate because of our locality optimization; most data is read from a local disk and bypasses our relatively bandwidth constrained network. The shuffle rate is higher than the output rate because the output phase writes two copies of the sorted data (we make two replicas of the output for reliability and availability reasons). We write two replicas because that is the mechanism for reliability and availability provided by our underlying file system. Network bandwidth requirements for writing data would be reduced if the underlying file system used erasure coding [15] rather than replication.

The original article has further experiments that examine the effects of backup tasks and machine failures [8].

6 Experience

We wrote the first version of the MapReduce library in February of 2003 and made significant enhancements to it in August of 2003, including the locality optimization, dynamic load balancing of task execution across worker machines, etc. Since that time, we have been pleasantly surprised at how broadly applicable the MapReduce library has been for the kinds of problems we work on. It has been used across a wide range of domains within Google, including:

- large-scale machine learning problems,
- clustering problems for the Google News and Froogle products,
- extracting data to produce reports of popular queries (e.g. Google Zeitgeist and Google Trends),
- extracting properties of Web pages for new experiments and products (e.g. extraction of geographical locations from a large corpus of Web pages for localized search),
- processing of satellite imagery data,
- language model processing for statistical machine translation, and
- large-scale graph computations.

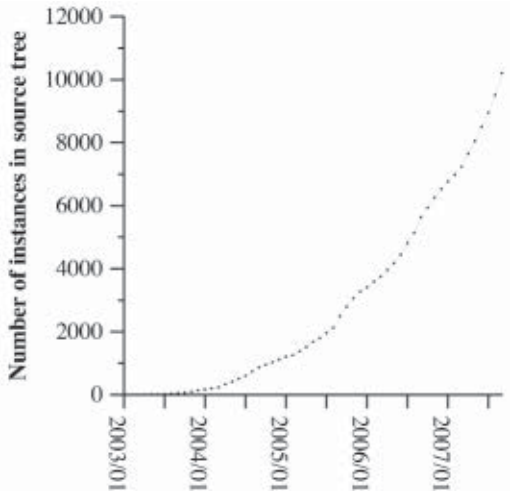


Fig. 4. MapReduce instances over time.

Figure 4 shows the significant growth in the number of separate MapReduce programs checked into our primary source-code management system over time, from 0 in early 2003 to almost 900 in September 2004, to about 4000 in March 2006. MapReduce has been so successful because it makes it possible to write a simple program and run it efficiently on a thousand machines in a half hour, greatly speeding up the development and prototyping cycle. Furthermore, it allows programmers who have no experience with distributed and/or parallel systems to exploit large amounts of resources easily.

Table I. MapReduce Statistics for Different Months.

	Aug. '04	Mar. '06	Sep. '07
Number of jobs (1000s)	29	171	2,217
Avg. completion time (secs)	634	874	395
Machine years used	217	2,002	11,081
map input data (TB)	3,288	52,254	403,152
map output data (TB)	758	6,743	34,774
reduce output data (TB)	193	2,970	14,018
Avg. machines per job	157	268	394
Unique implementations			
map	395	1958	4083
reduce	269	1208	2418

At the end of each job, the MapReduce library logs statistics about the computational resources used by the job. In Table I, we show some statistics for a subset of MapReduce jobs run at Google in various months, highlighting the extent to which MapReduce has grown and become the de facto choice for nearly all data processing needs at Google.

6.1 Large-Scale Indexing

One of our most significant uses of MapReduce to date has been a complete rewrite of the production indexing system that produces the data structures used for the Google Web search service. The indexing system takes as input a large set of documents that have been retrieved by our crawling system, stored as a set of GFS files. The raw contents for these documents are more than 20 terabytes of data. At the time we converted the indexing system to use MapReduce in 2003, it ran as a sequence of eight MapReduce operations. Since that time, because of the ease with which new phases can be added, many new phases have been added to the indexing system. Using MapReduce (instead of the ad-hoc distributed passes in the prior version of the indexing system) has provided several benefits.

- The indexing code is simpler, smaller, and easier to understand because the code that deals with fault tolerance, distribution, and parallelization is hidden within the MapReduce library. For example, the size of one phase of the computation dropped from approximately 3800 lines of C++ code to approximately 700 lines when expressed using MapReduce.
- The performance of the MapReduce library is good enough that we can keep conceptually unrelated computations separate instead of mixing them together to avoid extra passes over the data. This makes it easy to change the indexing process. For example, one change that took a few months to make in our old indexing system took only a few days to implement in the new system.

- The indexing process has become much easier to operate because most of the problems caused by machine failures, slow machines, and networking hiccups are dealt with automatically by the MapReduce library without operator intervention. Furthermore, it is easy to improve the performance of the indexing process by adding new machines to the indexing cluster.

7 Related Work

Many systems have provided restricted programming models and used the restrictions to parallelize the computation automatically. For example, an associative function can be computed over all prefixes of an N element array in log N time on N processors using parallel prefix computations [6, 11, 14]. MapReduce can be considered a simplification and distillation of some of these models based on our experience with large real-world computations. More significantly, we provide a fault-tolerant implementation that scales to thousands of processors. In contrast, most of the parallel processing systems have only been implemented on smaller scales and leave the details of handling machine failures to the programmer.

Our locality optimization draws its inspiration from techniques such as active disks [13, 17], where computation is pushed into processing elements that are close to local disks, to reduce the amount of data sent across I/O subsystems or the network.

The sorting facility that is a part of the MapReduce library is similar in operation to NOW-Sort [3]. Source machines (map workers) partition the data to be sorted and send it to one of R reduce workers. Each reduce worker sorts its data locally (in memory if possible). Of course NOW-Sort does not have the user-definable map and reduce functions that make our library widely applicable.

BAD-FS [5] and TACC [9] are two other systems that rely on reexecution as a mechanism for implementing fault tolerance.

The original article has a more complete treatment of related work [8].

Conclusions

The MapReduce programming model has been successfully used at Google for many different purposes. We attribute this success to several reasons. First, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault tolerance, locality optimization, and load balancing. Second, a large variety of problems are easily expressible as MapReduce computations. For example, MapReduce is used for the generation of data for Google's production Web search service, for sorting, data mining, machine learning, and many other systems. Third, we have developed an implementation of MapReduce that scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at Google.

By restricting the programming model, we have made it easy to parallelize and distribute computations and to make such computations fault tolerant. Second, network bandwidth is a scarce resource. A number of optimizations in our system are therefore targeted at reducing the amount of data sent across the network: the locality optimization allows us to read data from local disks, and writing a single copy of the intermediate data to local disk saves network bandwidth. Third, redundant execution can be used to reduce the impact of slow machines, and to handle machine failures and data loss.

Acknowledgements

Josh Levenberg has been instrumental in revising and extending the user-level MapReduce API with a number of new features. We would like to especially thank others who have worked on the system and all the users of MapReduce in Google's engineering organization for providing helpful feedback, suggestions, and bug reports.

References

1. Hadoop: Open source implementation of MapReduce. http://lucene.apache.org/hadoop/.
2. The Phoenix system for MapReduce programming. http://csl.stanford.edu/~christos/sw/phoenix/.
3. Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., Culler, D. E., Hellerstein, J. M., and Patterson, D. A. 1997. High-performance sorting on networks of workstations. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*. Tucson, AZ.
4 Barroso, L. A., Dean, J., and Urs Hölzle, U. 2003. Web search for a planet: The Google cluster architecture. *IEEE Micro* 23, 2, 22-28.
5. Bent, J., Thain, D., Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., and Livny, M. 2004. Explicit control in a batch-aware distributed file system. In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation* (*NSDI*).
6. Blelloch, G. E. 1989. Scans as primitive parallel operations. *IEEE Trans. Comput. C-38*, 11.
7. Chu, C.-T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G., Ng, A., and Olukotun, K. 2006. Map-Reduce for machine learning on multicore. In *Proceedings of Neural Information Processing Systems Conference* (*NIPS*). Vancouver, Canada.
8. Dean, J. and Ghemawat, S. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of Operating Systems Design and Implementation* (*OSDI*). San Francisco, CA. 137-150.
9. Fox, A., Gribble, S. D., Chawathe, Y., Brewer, E. A., and Gauthier, P. 1997. Cluster-based scalable network services. In *Proceedings of the 16th ACM Symposium on Operating System Principles*. Saint-Malo, France. 78-91.
10. Ghemawat, S., Gobioff, H., and Leung, S.-T. 2003. The Google file system. In *19th Symposium on Operating Systems Principles*. Lake George, NY. 29-43.
11. Gorlatch, S. 1996. Systematic efficient parallelization of scan and other list homomorphisms. In L. Bouge, P. Fraigniaud, A. Mignotte, and Y. Robert, Eds. *Euro-Par'96. Parallel Processing*, Lecture Notes in Computer Science, vol. 1124. Springer-Verlag. 401-408
12. Gray, J. Sort benchmark home page. http://research.microsoft.com/barc/SortBenchmark/.
13. Huston, L., Sukthankar, R., Wickremesinghe, R., Satyanarayanan, M., Ganger, G. R., Riedel, E., and Ailamaki, A. 2004. Diamond: A storage architecture for early discard in interactive search. In *Proceedings of the 2004 USENIX File and Storage Technologies FAST Conference*.
14. Ladner, R. E., and Fischer, M. J. 1980. Parallel prefix computation. *JACM* 27, 4. 831-838.
15. Rabin, M. O. 1989. Efficient dispersal of information for security, load balancing and fault tolerance. *JACM* 36, 2. 335-348.
16. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., and Kozyrakis, C. 2007. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of 13th International Symposium on High-Performance Computer Architecture* (*HPCA*). Phoenix, AZ.
17. Riedel, E., Faloutsos, C., Gibson, G. A., and Nagle, D. Active disks for large-scale data processing. *IEEE Computer*. 68-74.

TECHNICAL PERSPECTIVE:

FINDING A GOOD NEIGHBOR, NEAR AND FAST

by Bernard Chazelle

You haven't read it yet, but you can already tell this article is going to be one long jumble of words, numbers, and punctuation marks. Indeed, but look at it differently, as a text classifier would, and you will see a single point in high dimension, with word frequencies acting as coordinates. Or take the background on your flat panel display: a million colorful pixels teaming up to make quite a striking picture. Yes, but also one single point in 10^6-dimensional space—that is, if you think of each pixel's RGB intensity as a separate coordinate. In fact, you don't need to look hard to find complex, heterogeneous data encoded as clouds of points in high dimension. They routinely surface in applications as diverse as medical imaging, bioinformatics, astrophysics, and finance.

Why? One word: geometry. Ever since Euclid pondered what he could do with his compass, geometry has proven a treasure trove for countless computational problems. Unfortunately, high dimension comes at a price: the end of space partitioning as we know it. Chop up a square with two bisecting slices and you get four congruent squares. Now chop up a 100-dimensional cube in the same manner and you get 2^{100} little cubes—some Lego set! High dimension provides too many places to hide for searching to have any hope.

Just as dimensionality can be a curse (in Richard Bellman's words), so it can be a blessing for all to enjoy. For one thing, a multitude of random variables cavorting together tend to produce sharply concentrated measures: for example, most of the action on a high-dimensional sphere occurs near the equator, and any function defined over it that does not vary too abruptly is in fact nearly constant. For another blessing of dimensionality, consider Wigner's celebrated *semicircle law*: the spectral distribution of a large random matrix (an otherwise perplexing object) is described by a single, lowly circle. Sharp measure concentrations and easy spectral predictions are the foodstuffs on which science feasts.

But what about the curse? It can be vanquished. Sometimes. Consider the problem of storing a set S of n points in $\mathbf{R}^d$ (for very large d) in a data structure, so that, given any point q, the nearest $p \in S$ (in the Euclidean sense) can be found in a snap. Trying out all the points of S is a solution—a slow one. Another is to build the Voronoi diagram of S. This partitions $\mathbf{R}^d$ into regions with the same answers, so that handling a query q means identifying its relevant region. Unfortunately, any solution with the word "partition" in it is likely to raise the specter of the dreaded curse, and indeed this one lives up to that expectation. Unless your hard drive exceeds in bytes the number of particles in the universe, this "precompute and look up" method is doomed.

What if we instead lower our sights a little and settle for an approximate solution, say a point $p \in S$ whose distance to q is at most $c = 1 + \varepsilon$ times the smallest one? Luckily, in many applications (for example, data analysis, lossy compression, information retrieval, machine learning), the data is imprecise to begin with, so erring by a small factor of $c > 1$ does not cause much harm. And if it does, there is always the option (often useful in practice) to find the exact nearest neighbor by enumerating all points in the vicinity of the query: something the methods discussed below will allow us to do.

The pleasant surprise is that one can tolerate an arbitrarily small error and still break the curse. Indeed, a zippy query time of $O(d \log n)$ can be achieved with an amount of storage roughly $n^{O(\varepsilon^{-2})}$. No curse there. Only one catch: a relative error of, say, 10% requires a prohibitive amount of storage. So, while theoretically attractive, this solution and its variants have left practitioners unimpressed.

Enter Alexandr Andoni and Piotr Indyk [1], with a new solution that should appeal to theoretical and applied types alike. It is fast and economical, with software publicly available for slightly earlier incarnations of the method. The starting point is the classical idea of *locality-sensitive hashing* (LSH). The bane of classical hashing is collision: too many keys hashing to the same spot can ruin a programmer's day. LSH turns this weakness into a strength by hashing high-dimensional points into bins on a line in such a way that only nearby points collide. What better way to meet your neighbors than to bump into them? Andoni and Indyk modify LSH in critical ways to make neighbor searching more effective. For one thing, they hash down to spaces of logarithmic dimension, as opposed to single lines. They introduce a clever way of cutting up the hashing image space, all at a safe distance from the curse's reach. They also add bells and whistles from coding theory to make the algorithm more practical.

Idealized data structures often undergo cosmetic surgery on their way to industrial-strength implementations; such an evolution is likely in this latest form of LSH. But there is no need to wait for this. Should you need to find neighbors in very high dimension, one of the current LSH algorithms might be just the solution for you.

Biography

Bernard Chazelle (chazelle@cs.princeton.edu) is a professor of computer science at Princeton University, Princeton, NJ.

Reference

1. Andoni, A. and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on the Foundations of Computer Science* (FOCS'06).

Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions

by Alexandr Andoni and Piotr Indyk

Abstract

In this article, we give an overview of efficient algorithms for the approximate and exact nearest neighbor problem. The goal is to preprocess a dataset of objects (e.g., images) so that later, given a new query object, one can quickly return the dataset object that is most similar to the query. The problem is of significant interest in a wide variety of areas.

The goal of this article is twofold. In the first part, we survey a family of nearest neighbor algorithms that are based on the concept of *locality-sensitive hashing*. Many of these algorithm have already been successfully applied in a variety of practical scenarios. In the second part of this article, we describe a recently discovered hashing-based algorithm, for the case where the objects are points in the d-dimensional Euclidean space. As it turns out, the performance of this algorithm is provably near-optimal in the class of the locality-sensitive hashing algorithms.

1 Introduction

The *nearest neighbor* problem is defined as follows: given a collection of n points, build a data structure which, given any query point, reports the data point that is closest to the query. A particularly interesting and well-studied instance is where the data points live in a d-dimensional space under some (e.g., Euclidean) distance function. This problem is of major importance in several areas; some examples are data compression, databases and data mining, information retrieval, image and video databases, machine learning, pattern recognition, statistics and data analysis. Typically, the features of each object of interest (document, image, etc.) are represented as a point in $\mathbb{R}^d$ and the distance metric is used to measure the similarity of objects. The basic problem then is to perform indexing or similarity searching for query objects. The number of features (i.e., the dimensionality) ranges anywhere from tens to millions. For example, one can represent a 1000×1000 image as a vector in a 1,000,000-dimensional space, one dimension per pixel.

There are several efficient algorithms known for the case when the dimension d is low (e.g., up to 10 or 20). The first such data structure, called *kd-trees* was introduced in 1975 by Jon Bentley [6], and remains one of the most popular data structures used for searching in multidimensional spaces. Many other multidimensional data structures are known, see [35] for an overview. However, despite decades of intensive effort, the current solutions suffer from either space or query time that is *exponential* in d. In fact, for large enough d, in theory or in practice, they often provide little improvement over a linear time algorithm that compares a query to each point from the database. This phenomenon is often called "the curse of dimensionality."

In recent years, several researchers have proposed methods for overcoming the running time bottleneck by using approximation (e.g., [5, 27, 25, 29, 22, 28, 17, 13, 32, 1], see also [36, 24]). In this formulation, the algorithm is allowed to return a point whose distance from the query is at most c times the distance from the query to its nearest points; $c > 1$ is called the *approximation factor*. The appeal of this approach is that, in many cases, an approximate nearest neighbor is almost as good as the exact one. In particular, if the distance measure accurately captures the notion of user quality, then small differences in the distance should not matter. Moreover, an efficient approximation algorithm can be used to solve the exact nearest neighbor problem by enumerating all approximate nearest neighbors and choosing the closest point[1].

In this article, we focus on one of the most popular algorithms for performing approximate search in high dimensions based on the concept of *locality-sensitive hashing* (LSH) [25]. The key idea is to hash the points using several hash functions to ensure that for each function the probability of collision is much higher for objects that are close to each other than for those that are far apart. Then, one can determine near neighbors by hashing the query point and retrieving elements stored in buckets containing that point.

The LSH algorithm and its variants has been successfully applied to computational problems in a variety of areas, including web clustering [23], computational biology [10.11], computer vision (see selected articles in [23]), computational drug design [18] and computational linguistics [34]. A code implementing a variant of this method is available from the authors [2]. For a more theoretically-oriented overview of this and related algorithms, see [24].

The purpose of this article is twofold. In Section 2, we describe the basic ideas behind the LSH algorithm and its analysis; we also give an overview of the current library of LSH functions for various distance measures in Section 3. Then, in Section 4, we describe a recently developed LSH family for the Euclidean distance, which achievies a near-optimal separation between the collision probabilities of close and far points. An interesting feature of this family is that it effectively enables the reduction of the approximate nearest neighbor problem for worst-case data to the exact nearest neighbor problem over random (or pseudorandom) point configuration in low-dimensional spaces.

[1] See section 2.4 for more information about exact algorithms.

Biographies

Alexandr Andoni (andoni@mit.edu) is a Ph.D. Candidate in computer science at Massachusetts Institute of Technology, Cambridge, MA.

Piotr Indyk (indyk@theory.lcs.mit.edu) is an associate professor in the Theory of Computation Group, Computer Science and Artificial Intelligence Lab, at Massachusetts Institute of Technology, Cambridge, MA.

Currently, the new family is mostly of theoretical interest. This is because the asymptotic improvement in the running time achieved via a better separation of collision probabilities makes a difference only for a relatively large number of input points. Nevertheless, it is quite likely that one can design better pseudorandom point configurations which do not suffer from this problem. Some evidence for this conjecture is presented in [3], where it is shown that point configurations induced by so-called *Leech lattice* compare favorably with truly random configurations.

Preliminaries

2.1 Geometric Normed Spaces

We start by introducing the basic notation used in this article. First, we use P to denote the set of data points and assume that P has cardinality n. The points p from P belong to a d-dimensional space $\mathbb{R}^d$. We use p_i to the denote the ith coordinate of p, for $i = 1 \ldots d$.

For any two points p and q, the distance between them is defined as

$$\|p-q\|_s = \left(\sum_{i=1}^{d} |p_i - q_i|^s\right)^{1/s}$$

for a parameter $s > 0$; this distance function is often called the ℓ_s *norm*. The typical cases include $s = 2$ (the Euclidean distance) or $s = 1$ (the Manhattan distance)[2]. To simplify notation, we often skip the subscript 2 when we refer to the Euclidean norm, that is, $\|p-q\| = \|p-q\|_2$.

Occasionally, we also use the Hamming distance, which is defined as the number of positions on which the points p and q differ.

2.2 Problem Definition

The nearest neighbor problem is an example of an *optimization* problem: the goal is to find a point which minimizes a certain objective function (in this case, the distance to the query point). In contrast, the algorithms that are presented in this article solve the decision version of the problem. To simplify the notation, we say that a point p is an *R-near neighbor* of a point q if the distance between p and q is at most R (see Figure 1). In this language, our algorithm either returns one of the R-near neighbors or concludes that no such point exists for some parameter R.

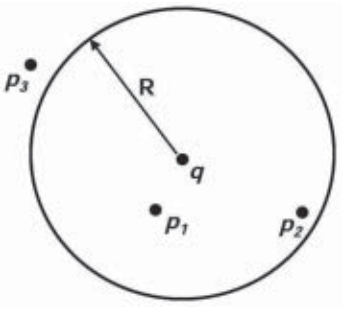


Fig. 1. An illustration of an* R*-near neighbor query. The nearest neighbor of the query point* q *is the point* p_1*. However, both* p_1 *and* p_2 *are* R*-near neighbors of* q*.

Naturally, the nearest and near neighbor problems are related. It is easy to see that the nearest neighbor problem also solves the R-near neighbor problem–one can simply check if the returned point is an R-near neighbor of the query point. The reduction in the other direction is somewhat more complicated and involves creating several instances of the near neighbor problem for different values of R. During the query time, the data structures are queried in the increasing order of R. The process is stopped when a data structure reports an answer. See [22] for a reduction of this type with theoretical guarantees.

In the rest of this article, we focus on the approximate near neighbor problem. The formal definition of the approximate version of the near neighbor problem is as follows.

Definition 2.1 (Randomized c-approximate R*-near neighbor, or* (c, R) – NN). Given a set P of points in a d-dimensional space $\mathbb{R}^d$, and parameters $R > 0$, $\delta > 0$, construct a data structure such that, given any query point q, if there exists an R-near neighbor of q in P, it reports some cR-near neighbor of q in P with probability $1 - \delta$.

For simplicity, we often skip the word randomized in the discussion. In these situations, we will assume that δ is an absolute constant bounded away from 1 (e.g., 1/2). Note that the probability of success can be amplified by building and querying several instances of the data structure. For example, constructing two independent data structures, each with $\delta = 1/2$, yields a data structure with a probability of failure $\delta = 1/2 \cdot 1/2 = 1/4$.

In addition, observe that we can typically assume that $R = 1$. Otherwise we can simply divide all coordinates by R. Therefore, we will often skip the parameter R as well and refer to the c-approximate near neighbor problem or c-NN.

We also define a related *reporting* problem.

Definition 2.2 (Randomized R*-near neighbor reporting).* Given a set P of points in a d-dimensional space $\mathbb{R}^d$, and parameters $R > 0$, $\delta > 0$, construct a data structure that, given any *query* point q, reports each R-near neighbor of q in P with probability $1 - \delta$.

Note that the latter definition does not involve an approximation factor. Also, unlike the case of the approximate near neighbor, here the data structure can return many (or even all) points if a large fraction of the data points are located close to the query point. As a result, one cannot give an a priori bound on the running time of the algorithm. However, as we point out later, the two problems are intimately related. In particular, the algorithms in this article can be easily modified to solve both c-NN and the reporting problems.

2.3 Locality-Sensitive Hashing

The LSH algorithm relies on the existence of *locality-sensitive hash functions*. Let $\mathcal{H}$ be a family of hash functions mapping $\mathbb{R}^d$ to some universe U. For any two points p and q, consider a process in which we choose a function h from $\mathcal{H}$ uniformly at random, and analyze the probability that $h(p) = h(q)$. The family $\mathcal{H}$ is called *locality sensitive* (with proper parameters) if it satisfies the following condition.

Definition 2.3 (Locality-sensitive hashing). A family $\mathcal{H}$ is called (R, cR, P_1, P_2)-sensitive if for any two points $p, q \in \mathbb{R}^d$.

- if $\|p-q\| \le R$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \ge P_1$,
- if $\|p-q\| \ge cR$ then $\Pr_{\mathcal{H}}[h(q) = h(p)] \le P_2$.

In order for a locality-sensitive hash (LSH) family to be useful, it has to satisfy $P_1 > P_2$.

[2]The name is motivated by the fact that $\|p-q\|_1 = \sum_{i=1}^{d} |p_i - q_i|$ is the length of the shortest path between p and q if one is allowed to move along only one coordinate at a time.

To illustrate the concept, consider the following example. Assume that the data points are *binary*, that is, each coordinate is either 0 or 1. In addition, assume that the distance between points p and q is computed according to the Hamming distance. In this case, we can use a particularly simple family of functions $\mathcal{H}$ which contains all projections of the input point on one of the coordinates, that is, $\mathcal{H}$ contains all functions h_i from $\{0, 1\}^d$ to $\{0, 1\}$ such that $h_i(p) = p_i$. Choosing one hash function h uniformly at random from $\mathcal{H}$ means that $h(p)$ returns a random coordinate of p (note, however, that different applications of h return the same coordinate of the argument).

To see that the family $\mathcal{H}$ is locality-sensitive with nontrivial parameters, observe that the probability $\Pr_{\mathcal{H}}[h(q) = h(p)]$ is equal to the fraction of coordinates on which p and q agree. Therefore, $P_1 = 1 - R/d$, while $P_2 = 1 - cR/d$. As long as the approximation factor c is greater than 1, we have $P_1 > P_2$.

2.4 The Algorithm

An LSH family $\mathcal{H}$ can be used to design an efficient algorithm for approximate near neighbor search. However, one typically cannot use $\mathcal{H}$ as is since the gap between the probabilities P_1 and P_2 could be quite small. Instead, an amplification process is needed in order to achieve the desired probabilities of collision. We describe this step next, and present the complete algorithm in the Figure 2.

Given a family $\mathcal{H}$ of hash functions with parameters (R, cR, P_1, P_2) as in Definition 2.3, we amplify the gap between the high probability P_1 and low probability P_2 by concatenating several functions. In particular, for parameters k and L (specified later), we choose L functions $g_j(q) = (h_{1,j}(q),\ldots,h_{k,j}(q))$, where $h_{t,j}$ $(1 \le t \le k,\ 1 \le j \le L)$ are chosen independently and uniformly at random from $\mathcal{H}$. These are the actual functions that we use to hash the data points.

The data structure is constructed by placing each point p from the input set into a bucket $g_j(p)$, for $j = 1,\ldots,L$. Since the total number of buckets may be large, we retain only the nonempty buckets by resorting to (standard) hashing[3] of the values $g_j(p)$. In this way, the data structure uses only $O(nL)$ memory cells; note that it suffices that the buckets store the pointers to data points, not the points themselves.

To process a query q, we scan through the buckets $g_1(q),\ldots, g_L(q)$, and retrieve the points stored in them. After retrieving the points, we compute their distances to the query point, and report any point that is a valid answer to the query. Two concrete scanning strategies are possible.

1. Interrupt the search after finding the first L' points (including duplicates) for some parameter L'.
2. Continue the search until all points from all buckets are retrieved; no additional parameter is required.

The two strategies lead to different behaviors of the algorithms. In particular, Strategy 1 solves the (c, R)-near neighbor problem, while Strategy 2 solves the R-near neighbor reporting problem.

Strategy 1. It is shown in [25, 19] that the first strategy, with $L' = 3L$, yields a solution to the *randomized c-approximate R-near neighbor problem*, with parameters R and δ for some constant failure probability $\delta < 1$. To obtain this guarantee, it suffices to set L to $\Theta(n^\rho)$, where $\rho = \frac{\ln 1/P_1}{\ln 1/P_2}$ [19]. Note that this implies that the algorithm runs in time proportional to n^ρ which is sublinear in n if $P_1 > P_2$. For example, if we use the hash functions for the binary vectors mentioned earlier, we obtain $\rho = 1/c$ [25, 19]. The exponents for other LSH families are given in Section 3.

Strategy 2. The second strategy enables us to solve the *randomized R-near neighbor reporting problem*. The value of the failure probability δ depends on the choice of the parameters k and L. Conversely, for each δ, one can provide parameters k and L so that the error probability is smaller than δ. The query time is also dependent on k and L. It could be as high as $\Theta(n)$ in the worst case, but, for many natural datasets, a proper choice of parameters results in a sublinear query time.

The details of the analysis are as follows. Let p be any R-neighbor of q, and consider any parameter k. For any function g_i, the probability that $g_i(p) = g_i(q)$ is at least P_1^k. Therefore, the probability that $g_i(p) = g_i(q)$ for *some* $i = 1\ldots L$ is at least $1 - (1 - P_1^k)^L$. If we set $L = \log_{1-P_1^k} \delta$ so that $(1 - P_1^k)^L \le \delta$, then any R-neighbor of q is returned by the algorithm with probability at least $1 - \delta$.

How should the parameter k be chosen? Intuitively, larger values of k lead to a larger gap between the probabilities of collision for close points and far points; the probabilities are P_1^k and P_2^k, respectively (see Figure 3 for an illustration). The benefit of this amplification is that the hash functions are more selective. At the same time, if k is large then P_1^k is small, which means that L must be sufficiently large to ensure that an R-near neighbor collides with the query point at least once.

[3] See [16] for more details on hashing.

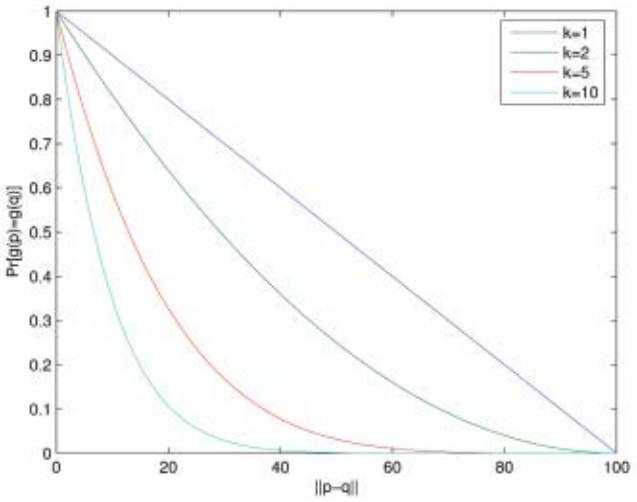
Preprocessing:

1. Choose L functions g_j, $j = 1,\ldots L$, by setting $g_j = (h_{1,j}, h_{2,j},\ldots h_{k,j})$, where $h_{1,j},\ldots h_{k,j}$ are chosen at random from the LSH family $\mathcal{H}$.
2. Construct L hash tables, where, for each $j = 1,\ldots L$, the j^{th} hash table contains the dataset points hashed using the function g_j.

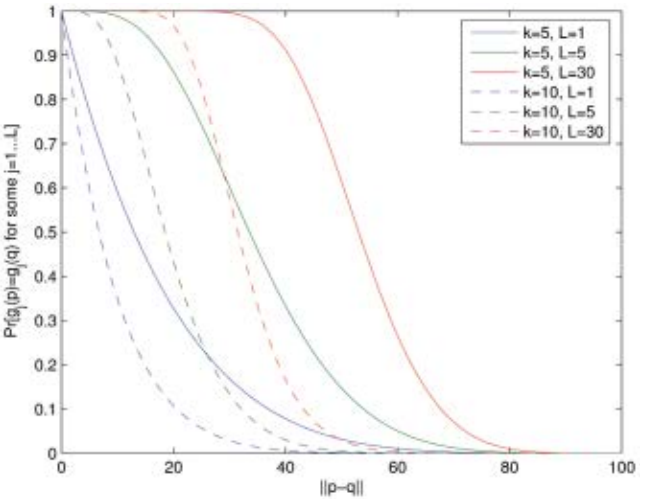
Query algorithm for a query point q:

1. For each $j = 1, 2,\ldots L$
   i) Retrieve the points from the bucket $g_j(q)$ in the j^{th} hash table.
   ii) For each of the retrieved point, compute the distance from q to it, and report the point if it is a correct answer (cR-near neighbor for Strategy 1, and R-near neighbor for Strategy 2).
   iii) (*optional*) Stop as soon as the number of reported points is more than L'.

Fig. 2. Preprocessing and query algorithms of the basic LSH algorithm.



(a) The probability that $g_j(p) = g_j(q)$ for a fixed j. Graphs are shown for several values of k. In particular, the blue function ($k = 1$) is the probability of collision of points p and q under a single random hash function h from the LSH family.



(b) The probability that $g_j(p) = g_j(q)$ for some $j = 1\ldots L$. The probabilities are shown for two values of k and several values of L. Note that the slopes are sharper when k is higher.

Fig. 3. The graphs of the probability of collision of points p ***and*** q ***as a function of the distance between*** p ***and*** q ***for different values of*** k ***and*** L. ***The points*** p ***and*** q ***are*** d = *100 dimensional binary vectors under the Hamming distance. The LSH family $\mathcal{H}$ is the one described in Section 2.3.***

A practical approach to choosing k was introduced in the E^{2}LSH package [2]. There the data structure optimized the parameter k as a function of the dataset and a set of sample queries. Specifically, given the dataset, a query point, and a fixed k, one can estimate precisely the expected number of collisions and thus the time for distance computations as well as the time to hash the query into all L hash tables. The sum of the estimates of these two terms is the estimate of the total query time for this particular query. E^{2}LSH chooses k that minimizes this sum over a small set of sample queries.

3 LSH Library

To date, several LSH families have been discovered. We briefly survey them in this section. For each family, we present the procedure of chosing a random function from the respective LSH family as well as its locality-sensitive properties.

Hamming distance. For binary vectors from $\{0, 1\}^d$, Indyk and Motwani [25] propose LSH function $h_i(p) = p_i$, where $i \in \{1,\ldots d\}$ is a randomly chosen index (the sample LSH family from Section 2.3). They prove that the exponent ρ is $1/c$ in this case.

It can be seen that this family applies directly to M-ary vectors (i.e., with coordinates in $\{1\ldots M\}$) under the Hamming distance. Moreover, a simple reduction enables the extension of this family of functions to M-ary vectors under the ℓ_1 distance [30]. Consider any point p from $\{1\ldots M\}^d$. The reduction proceeds by computing a binary string $Unary(p)$ obtained by replacing each coordinate p_i by a sequence of p_i ones followed by $M - p_i$ zeros. It is easy to see that for any two M-ary vectors p and q, the Hamming distance between $Unary(p)$ and $Unary(p)$ equals the ℓ_1 distance between p and q. Unfortunately, this reduction is efficient only if M is relatively small.

ℓ_1 distance. A more direct LSH family for $\mathbb{R}^d$ under the ℓ_1 distance is described in [4]. Fix a real $w \gg R$, and impose a randomly shifted grid with cells of width w; each cell defines a bucket. More specifically, pick random reals $s_1, s_2,\ldots s_d \in [0, w)$ and define $h_{s_1,\ldots s_d} = (\lfloor (x_1 - s_1)/w \rfloor,\ldots,\lfloor (x_d - s_d)/w \rfloor)$. The resulting exponent is equal to $\rho = 1/c + O(R/w)$.

ℓ_s distance. For the Euclidean space, [17] propose the following LSH family. Pick a random projection of $\mathbb{R}^d$ onto a 1-dimensional line and chop the line into segments of length w, shifted by a random value $b \in [0, w)$. Formally, $h_{r,b} = \lfloor (r \cdot x + b)/w \rfloor$, where the projection vector $r \in \mathbb{R}^d$ is constructed by picking each coordinate of r from the Gaussian distribution. The exponent ρ drops strictly below $1/c$ for some (carefully chosen) finite value of w. This is the family used in the [2] package.

A generalization of this approach to ℓ_s norms for any $s \in [0, 2)$ is possible as well; this is done by picking the vector r from so-called *s-stable distribution*. Details can be found in [17].

Jaccard. To measure the similarity between two sets $A, B \subset U$ (containing, e.g., words from two documents), the authors of [9, 8] utilize the *Jaccard coefficient*. The Jaccard coefficient is defined as $s(A, B) = \frac{|A \cap B|}{|A \cup B|}$. Unlike the Hamming distance, Jaccard coefficient is a similarity measure: higher values of Jaccard coefficient indicate higher similarity of the sets. One can obtain the corresponding distance measure by taking $d(A, B) = 1 - s(A, B)$. For this measure, [9, 8] propose the following LSH family, called *min-hash*. Pick a random permutation on the ground universe U. Then, define $h_\pi(A) = \min\{\pi(a) \mid a \in A\}$. It is not hard to prove that the probability of collision $\Pr_\pi[h_\pi(A) = h_\pi(B)] = s(A, B)$. See [7] for further theoretical developments related to such hash functions.

Arccos. For vectors $p, q \in \mathbb{R}^d$, consider the distance measure that is the angle between the two vectors, $\theta(p, q) = \arccos\left(\frac{p \cdot q}{\|p\| \cdot \|q\|}\right)$. For this distance measure, Charikar et al. (inspired by [20]) defines the following LSH family [14]. Pick a random unit-length vector $u \in \mathbb{R}^d$ and define $h_u(p) = \text{sign}(u \cdot p)$. The hash function can also be viewed as partitioning the space into two half-spaces by a randomly chosen hyperplane. Here, the probability of collision is $\Pr_u[h_u(p) = h_u(q)] = 1 - \theta(p, q)/\pi$.

ℓ_2 distance on a sphere. Terasawa and Tanaka [37] propose an LSH algorithm specifically designed for points that are on a unit hypersphere in the Euclidean space. The idea is to consider a regular polytope, orthoplex for example, inscribed into the hypersphere and rotated at random. The hash function then maps a point on the hypersphere into the closest polytope vertex lying on the hypersphere. Thus, the buckets of the hash function are the Voronoi cells of the polytope vertices lying on the hypersphere. [37] obtain exponent ρ that is an improvement over [17] and the Leech lattice approach of [3].

4 Near-Optimal LSH Functions for Euclidean Distance

In this section we present a new LSH family, yielding an algorithm with query time exponent $\rho(c) = 1/c^2 + O(\log \log n / \log^{1/3} n)$. For large enough n, the value of $\rho(c)$ tends to $1/c^2$. This significantly improves upon the earlier running time of [17]. In particular, for $c = 2$, our exponent tends to 0.25, while the exponent in [17] was around 0.45. Moreover, a recent paper [31] shows that hashing-based algorithms (as described in Section 2.3) cannot achieve $\rho < 0.462/c^2$. Thus, the running time exponent of our algorithm is essentially optimal, up to a constant factor.

We obtain our result by carefully designing a family of locality-sensitive hash functions in ℓ_2. The starting point of our construction is the line partitioning method of [17]. There, a point p was mapped into $\mathbb{R}^1$ using a random projection. Then, the line $\mathbb{R}^1$ was partitioned into intervals of length w, where w is a parameter. The hash function for p returned the index of the interval containing the projection of p.

An analysis in [17] showed that the query time exponent has an interesting dependence on the parameter w. If w tends to infinity, the exponent tends to $1/c$, which yields no improvement over [25, 19]. However, for small values of w, the exponent lies slightly below $1/c$. In fact, the unique minimum exists for each c.

In this article, we utilize a "multi-dimensional version" of the aforementioned approach. Specifically, we first perform random projection into $\mathbb{R}^t$, where t is super-constant, but relatively small (i.e., $t = o(\log n)$). Then we partition the space $\mathbb{R}^t$ into cells. The hash function function returns the index of the cell which contains projected point p.

The partitioning of the space $\mathbb{R}^t$ is somewhat more involved than its one-dimensional counterpart. First, observe that the natural idea of partitioning using a grid does not work. This is because this process roughly corresponds to hashing using concatenation of several one-dimensional functions (as in [17]). Since the LSH algorithms perform such concatenation anyway, grid partitioning does not result in any improvement. Instead, we use the method of "ball partitioning", introduced in [15], in the context of embeddings into tree metrics. The partitioning is obtained as follows. We create a sequence of balls $B_1, B_2 \ldots$, each of radius w, with centers chosen independently at random. Each ball B_i then defines a cell, containing points $B_i \setminus \cup_{j<i} B_j$.

In order to apply this method in our context, we need to take care of a few issues. First, locating a cell containing a given point could require enumeration of all balls, which would take an unbounded amount of time. Instead, we show that one can simulate this procedure by replacing each ball by a grid of balls. It is not difficult then to observe that a finite (albeit exponential in t) number U of such grids suffices to cover all points in $\mathbb{R}^t$. An example of such partitioning (for $t = 2$ and $U = 5$) is given in Figure 4.

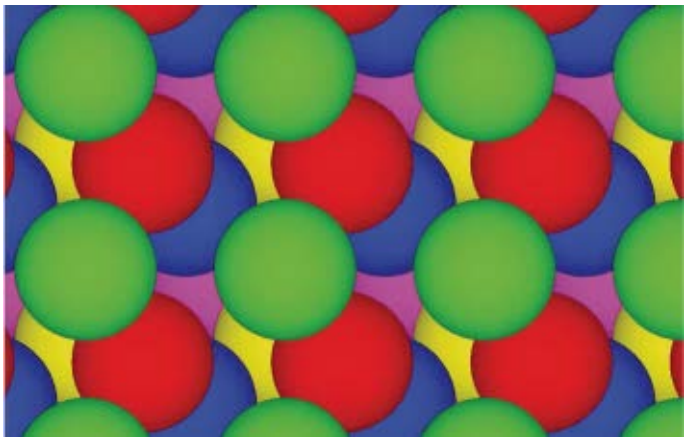


Fig. 4. An illustration of the the ball partitioning of the 2-dimensional space.

The second and the main issue is the choice of w. Again, it turns out that for large w, the method yields only the exponent of $1/c$. Specifically, it was shown in [15] that for any two points $p, q \in \mathbb{R}^t$, the probability that the partitioning separates p and q is at most $O(\sqrt{t} \cdot \|p - q\|/w)$. This formula can be showed to be tight for the range of w where it makes sense as a lower bound, that is, for $w = \Omega(\sqrt{t} \cdot \|p - q\|)$. However, as long as the separation probability depends linearly on the distance between p and q, the exponent ρ is still equal to $1/c$. Fortunately, a more careful analysis[4] shows that, as in the one-dimensional case, the minimum is achieved for finite w. For that value of w, the exponent tends to $1/c^2$ as t tends to infinity.

5 Related Work

In this section, we give a brief overview of prior work in the spirit of the algorithms considered in this article. We give only high-level simplified descriptions of the algorithms to avoid area-specific terminology. Some of the papers considered a closely related problem of finding all close pairs of points in a dataset. For simplicity, we translate them into the near neighbor framework since they can be solved by performing essentialy n separate near neighbor queries.

Hamming distance. Several papers investigated multi-index hashing-based algorithms for retrieving similar pairs of vectors with respect to the Hamming distance. Typically, the hash functions were projecting the vectors on some subset of the coordinates $\{1 \ldots d\}$ as in the example from an earlier section. In some papers [33, 21], the authors considered the probabilistic model where the data points are chosen uniformly at random, and the query point is a random point close to one of the points in the dataset. A different approach [26] is to assume that the dataset is arbitrary, but almost all points are far from the query point. Finally, the paper [12] proposed an algorithm which did not make any assumption on the input. The analysis of the algorithm was akin to the analysis sketched at the end of section 2.4: the parameters k and L were chosen to achieve desired level of sensitivity and accuracy.

Set intersection measure. To measure the similarity between two sets A and B, the authors of [9, 8] considered the Jaccard coefficient $s(A, B)$, proposing a family of hash functions $h(A)$ such that $\Pr[h(A) = h(B)] = s(A, B)$ (presented in detail in Section 3). Their main motivation was to

[4] Refer to [3] for more details.

construct short similarity-preserving "sketches" of sets, obtained by mapping each set A to a sequence $\langle h_1(A), ..., h_k(A)\rangle$. In section 5.3 of their paper, they briefly mention an algorithm similar to Strategy 2 described at the end of the Section 2.4. One of the differences is that, in their approach, the functions h_i are sampled without replacement, which made it more difficult to handle small sets.

Acknowledgement

This work was supported in part by NSF CAREER grant CCR-0133849 and David and Lucille Packard Fellowship.

References

1. Ailon, N. and Chazelle, B. 2006. Approximate nearest neighbors and the Fast Johnson-Lindenstrauss Transform. In *Proceedings of the Symposium on Theory of Computing*.
2. Andoni, A. and Indyk, P. 2004. E2lsh: Exact Euclidean locality-sensitive hashing. http://web.mit.edu/andoni/www/LSH/.
3. Andoni, A. and Indyk, P. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the Symposium on Foundations of Computer Science*.
4. Andoni, A. and Indyk, P. 2006. Efficient algorithms for substring near neighbor problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*. 1203–1212.
5. Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. 1994. An optimal algorithm for approximate nearest neighbor searching. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*. 573–582.
6. Bentley, J. L. 1975. Multidimensional binary search trees used for associative searching. *Comm. ACM 18*, 509–517.
7. Broder, A., Charikar, M., Frieze, A., and Mitzenmacher, M. 1998. Min-wise independent permutations. *J. Comput. Sys. Sci.*
8. Broder, A., Glassman, S., Manasse, M., and Zweig, G. 1997. Syntactic clustering of the web. In *Proceedings of the 6th International World Wide Web Conference*. 391–404.
9. Broder, A. 1997. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences*. 21–29.
10. Buhler, J. 2001. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinform. 17*, 419–428.
11. Buhler, J. and Tompa, M. 2001. Finding motifs using random projections. In *Proceedings of the Annual International Conference on Computational Molecular Biology* (*RECOMB1*).
12. Califano, A. and Rigoutsos, I. 1993. Flash: A fast look-up algorithm for string homology. In *Proceedings of the IEE Conference on Computer Vision and Pattern Recognition* (*CVPR*).
13. Chakrabarti, A. and Regev, O. 2004. An optimal randomised cell probe lower bounds for approximate nearest neighbor searching. In *Proceedings of the Symposium on Foundations of Computer Science*.
14. Charikar, M. 2002. Similarity estimation techniques from rounding. In *Proceedings of the Symposium on Theory of Computing*.
15. Charikar, M., Chekuri, C., Goel, A., Guha, S., and Plotkin, S. 1998. Approximating a finite metric by a small number of tree metrics. In *Proceedings of the Symposium on Foundations of Computer Science*.
16. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. 2001. *Introduct. Algorithms*. 2nd Ed. MIT Press.
17. Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V. 2004. Locality-sensitive hashing scheme based on p-stable distributions.In *Proceedings of the ACM Symposium on Computational Geometry*.
18. Dutta, D., Guha, R., Jurs, C., and Chen, T. 2006. Scalable partitioning and exploration of chemical spaces using geometric hashing. *J. Chem. Inf. Model. 46*.
19. Gionis, A., Indyk, P., and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Databases*.
20. Goemans, M. and Williamson, D. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM 42*. 1115–1145.
21. Greene, D., Parnas, M., and Yao, F. 1994. Multi-index hashing for information retrieval. In Proceedings of the Symposium on Foundations of Computer Science. 722–731.
22. Har-Peled, S. 2001. A replacement for voronoi diagrams of near linear size. In *Proceedings of the Symposium on Foundations of Computer Science*.
23. Haveliwala, T., Gionis, A., and Indyk, P. 2000. Scalable techniques for clustering the web. *WebDB Workshop*.
24. Indyk, P. 2003. Nearest neighbors in high-dimensional spaces. In *Handbook of Discrete and Computational Geometry*. CRC Press.
25. Indyk, P. and Motwani, R. 1998. Approximate nearest neighbor: Towards removing the curse of dimensionality. In *Proceedings of the Symposium on Theory of Computing*.
26. Karp, R. M., Waarts, O., and Zweig, G. 1995. The bit vector intersection problem. In *Proceedings of the Symposium on Foundations of Computer Science*. pages 621–630.
27. Kleinberg, J. 1997. Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the Symposium on Theory of Computing*.
28. Krauthgamer, R. and Lee, J. R. 2004. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*.
29. Kushilevitz, E., Ostrovsky, R., and Rabani, Y. 1998. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the Symposium on Theory of Computing*. 614–623.
30. Linial, N., London, E., and Rabinovich, Y. 1994. The geometry of graphs and some of its algorithmic applications. In *Proceedings of the Symposium on Foundations of Computer Science*. 577–591.
31. Motwani, R., Naor, A., and Panigrahy, R. 2006. Lower bounds on locality sensitive hashing. In *Proceedings of the ACM Symposium on Computational Geometry*.
32. Panigrahy, R. 2006. Entropy-based nearest neighbor algorithm in high dimensions. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*.
33. Paturi, R., Rajasekaran, S., and Reif, J.The light bulb problem. *Inform. Comput. 117*, 2, 187–192.
34. Ravichandran, D., Pantel, P., and Hovy, E. 2005. Randomized algorithms and nlp: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*.
35. Samet, H. 2006. *Foundations of Multidimensional and Metric Data Structures*. Elsevier, 2006.
36. Shakhnarovich, G., Darrell, T., and Indyk, P. Eds. *Nearest Neighbor Methods in Learning and Vision*. Neural Processing Information Series, MIT Press.
37. Terasawa, T. and Tanaka, Y. 2007. Spherical lsh for approximate nearest neighbor search on unit hypersphere. In *Proceedings of the Workshop on Algorithms and Data Structures*.

Group Term Life Insurance**

10- or 20-Year Group Term Life Insurance*

Group Disability Income Insurance*

Group Accidental Death & Dismemberment Insurance*

Group Catastrophic Major Medical Insurance*

Group Dental Plan*

Long-Term Care Plan

Major Medical Insurance

Short-Term Medical Plan***

Who has time to think about insurance?

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

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

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

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

MARSH
Affinity Group Services
a service of Seabury & Smith

eLearn MAGAZINE

Education and Technology in Perspective

elearnmag.org

Arizona State University
Department of Mathematical Sciences and Applied Computing

Assistant Professor of Applied Computing, Department of Mathematical Sciences and Applied Computing, Arizona State University. Full-time tenure track position beginning August 16, 2008. We are seeking applicants to conduct research in computer science, to develop and teach new undergraduate courses, and to forge interdisciplinary collaborations in research and/or teaching. Evidence of expertise in database systems or network and distributed processing is required. For complete application information and requirements see http://newcollege.asu.edu/dean/employment/msac_faculty.shtml. Application deadline is January 4, 2008; if not filled, weekly thereafter until the search is closed. AA/EOE.

Armstrong Atlantic State University
Savannah, Georgia
Department Head, Department of Information Technology

Armstrong Atlantic State University invites applications for the position of Department Head of the Department of Information Technology. The Head of the Department of Information Technology reports to the Dean of the College and is required to have a doctoral degree in Information Technology or related field, demonstrated administrative and leadership experience, evidence of a strong commitment to undergraduate education, and a strong record of scholarship. The Department of Information Technology at Armstrong Atlantic has approximately 140 information technology majors in the on-campus undergraduate Bachelor of Information Technology program. Additionally, the department is part of a consortium with five other state universities offering an on-line Bachelor of Science in Information Technology degree (WebBSIT). The department is completing work towards the creation of a new interdisciplinary minor in cyber security. IT faculty have been awarded several substantial externally funded grants that support students, either through scholarships or stipends. Please visit the AASU Human Resources website, www.hr.armstrong.edu/jobs.htm for additional details about this position, including application procedures and contact information. For full consideration, application materials must be received by January 21st, 2008. Review of applications will begin January 21st, 2008 and will continue until the position is filled. Employment is contingent upon successful completion of a background investigation.

Georgia is an Open Records Law state
AA/EOE

Austin Peay State University
Department of Computer Science

The Dept of Comp Sci & Info Tech at Austin Peay State University invites applications for a tenure-track assistant professor position, contingent upon funding, in computer science beginning August 2008. For more information, see http://www.apsu.edu/faculty/positions/index.htm.

Berea College
Mathematics and Computer Science Department

Berea College announces a full time, tenure-track position in the Mathematics and Computer Science Department, beginning September, 2008. Appointment will be at the assistant professor level. A Ph.D. in Computer Science and willingness to teach courses in mathematics or a Ph.D. in the mathematical sciences with a willingness to teach courses in computer science is required. A strong commitment to teaching is essential. Responsibilities center on mathematics and computer science teaching ranging from introductory to advanced undergraduate. Above all we are seeking candidates who can achieve excellence in teaching and who, in an undergraduate environment, will find ways to grow professionally. All faculty in the Department will be expected to interact with students on a one-on-one basis in the excitement and vitality of their growth through such activities as summer faculty/student research, independent studies, or senior capstone projects. The Department is supportive of all forms of scholarship. Applicants should send a cover letter, resume, transcripts of graduate and undergraduate work, a statement of personal teaching philosophy, and three letters of recommendation by January 21st to Professor James Blackburn-Lynch, Chair, Mathematics and Computer Science Department, CPO 2146, Berea College, Berea, KY 40404. More information about Berea College and the Mathematics and Computer Science Department is available at http://www.berea.edu/mcs/ Women and minority candidates are especially encouraged to apply.

Berea College, in light of its mission in the tradition of impartial love and social equality, welcomes all people of the earth to learn and work here.

Boston University
Department of Electrical and Computer Engineering

The Department of Electrical and Computer Engineering (ECE) at Boston University anticipates openings for faculty positions at all ranks in all areas of electrical and computer engineering. Areas of particular interest in

PAUL WATSON

computer engineering are computer systems, embedded and reconfigurable systems, distributed systems, trusted computing, design automation, VLSI, computer networks, software engineering, and related areas.

The ECE Department is part of a rapidly developing and innovative College of Engineering. Excellent opportunities exist for collaboration with colleagues in outstanding research centers at Boston University, at other universities/colleges, and with industry throughout the Boston area. The Department has 44 faculty members, 200 graduate students and 250 BS majors. For additional information, please visit http://www.bu.edu/ece/.

In addition to a relevant, earned PhD, qualified candidates will have a demonstrable ability to teach effectively, develop funded research programs in their area of expertise, and contribute to the tradition of excellence in research that is characteristic of the ECE department.

Applicants should send their curriculum vita with a statement of teaching and research plans to: Professor David Castañón, Chair ad interim, Department of Electrical and Computer Engineering, Boston University, 8 Saint Mary's Street, Boston, MA 02215. Boston University is an Equal Opportunity/Affirmative Action Employer.

Cal Poly, San Luis Obispo
Computer Science Department

COMPUTER SCIENCE - Full-time academic year tenure track faculty positions available in the Computer Science Department at Cal Poly, San Luis Obispo, California, beginning September 8, 2008. Rank and salary is commensurate with qualifications and experience. Duties include teaching core undergraduate courses, and upper-division and master's level courses in a specialty area; performing research in a mainstream area of computer science; and service to the department, the university, and the community. Applicants from all mainstream areas of computer science and software engineering are encouraged to apply. A doctorate in Computer Science, Software Engineering or a closely related field is required. Candidates must have a strong commitment to teaching excellence and laboratory-based instruction; dedication to continued professional development and scholarship; and a broad-based knowledge of computer science. Demonstrated ability in the written and oral use of the English language is required. Cal Poly offers BS and MS degrees in Computer Science, BS in Software Engineering, and a BS in Computer Engineering. Cal Poly emphasizes "learn by doing" which involves extensive lab work and projects in support of theoretical knowledge. The available computing facilities for instructional and faculty support are modern and extensive. To apply, please visit WWW.CALPOLYJOBS.ORG and complete a required online faculty application and apply to Requisition #101386. Review of applications will begin January 7, 2008; applications received after that date may be considered. For full consideration, candidates are required to attach to their online application: (1) resume, (2) cover letter, (3) statement of goals and plans for teaching and research. Candidates selected as a finalist will be required to submit three letters of reference and official transcripts for final consideration. Questions can be emailed to: csc-recruit@csc.calpoly.edu. Please include requisition #101386 in all correspondence. For further information about the department and its programs, see www.csc.calpoly.edu. Cal Poly is strongly committed to achieving excellence through cultural diversity. The university actively encourages applications and nominations of all qualified individuals. EEO.

Carnegie Mellon University
Faculty Positions in the Human-Computer Interaction Institute at Carnegie Mellon

Our world-class interdisciplinary Human-Computer Interaction Institute at Carnegie Mellon University (http://www.hcii.cmu.edu/) expects to fill one or more tenure-track faculty positions starting 8/08.

Design Research and Human Computer Interaction We are especially interested in an extraordinary faculty member who will advance our interdisciplinary research in design dimensions of human-computer interaction. We expect to hire at any level including senior level.

Learning Science and Educational Technology We seek a faculty member (assistant professor level) who will significantly advance our interdisciplinary work in learning science and technology research in intelligent tutoring systems, laboratory and classroom experimentation, educational data mining, computer-supported collaborative learning, ubiquitous computing in formal and informal learning settings, and educational design research.

Human Computer Interaction The area is open but we are especially seeking candidates at the assistant professor level whose work will advance our interdisciplinary research in social computing and the application of innovative methods (such as visualization, AI, and so forth) to HCI.

Applicants for a position should have the terminal degree in a discipline such as (but not limited to) design, computer science, psychology, HCI, or cognitive science. We seek

an outstanding educator and researcher who designs systems, implements systems, and/or performs rigorous empirical laboratory or qualitative field studies. The candidate must be able to significantly advance research and theory in his/her own field and HCI.

Review of faculty applications will begin December 15. Your application should include your CV, webpage URL, a statement of your research and teaching interests, copies of 1-3 representative papers, and the names, positions, and email addresses of three or more individuals who may be asked to provide letters of reference. Indicate your U.S. citizenship or describe your current visa status. Please send to Faculty Search Committee, hcii_facultysearch@cs.cmu.edu.

Carnegie Mellon is an affirmative action/equal opportunity employer and we invite and encourage applications from women and minorities.

CastTV
Software Engineer

CastTV in San Francisco Software Engineer Fascinated by challenging machine learning, distributed computing, web crawling and search problems? Were hiring entrepreneurial engineers to build the webs best video search. email: jobs@casttv.com

Connecticut College, Trinity College and Wesleyan University
Computer Science Department

The CTW Consortium (Connecticut College, Trinity College and Wesleyan University) announces a two-year postdoctoral fellowship in Computer Science funded by the Andrew W. Mellon Foundation to begin in August 2008. The reduced teaching load of one course per semester ensures ample time for research. Applicants should have received a Ph.D. in Computer Science or related field in the last 5 years. Responsibilities will include conducting research, preparing new courses, giving seminar talks, and involving undergraduates in research. Review of applications will start immediately. Candidates should send a letter describing their teaching and research interests, a current vita, and three letters of reference to Professor Gary Parker at: parker@conncoll.edu. All three institutions are Affirmative Action/Equal Opportunity Employers. For further information see:http://cs.conncoll.edu/mellonfellows.html

Drexel University
Department of Computer Science

Drexel University's Department of Computer Science (www.cs.drexel.edu) invites applications for tenure-track faculty positions at all levels. The preferred interest is ARTIFICIAL INTELLIGENCE and MULTI-AGENT SYSTEMS, although exceptional applicants in other areas will be considered. The department has expanding graduate research and education programs in software engineering, graphics and vision, information assurance and security, human-computer interaction, high-performance computing and symbolic computation. We specialize in interdisciplinary and applied research and are supported by several major federal research grants from NSF, DARPA, ONR, DoD, DoE and NIST, as well as by private sources such as Intel, Nissan, NTT, and Lockheed Martin. The department offers BS, BA, MS, and Ph.D. degrees in computer science as well as BS and MS degrees in software engineering. Drexel is a designated National Security Agency (NSA) Center of Academic Excellence in Information Assurance Education. The department has over 425 undergraduate and over 100 graduate students, with annual research expenditures in excess of $4M. Several of the Computer Science faculty are recipients of NSF CAREER or Young Investigator Awards. Review of applications begins im-

Max Planck Institute for Software Systems

Tenure-track openings

Applications are invited for tenure-track and tenured positions in all areas related to the design, analysis and engineering of software systems, including programming languages, formal methods, security, distributed, networked and embedded systems, databases and information systems, and human-computer interaction. A doctoral degree in computer science or related areas and an outstanding research record are required. Successful candidates are expected to build a team and pursue a highly visible research agenda, both independently and in collaboration with other groups. Senior candidates must have demonstrated leadership abilities and recognized international stature.

The institute offers a unique environment that combines the best aspects of a university department and a research laboratory:
- Successful candidates receive generous base funding to build and lead a team of graduate students and post-docs. They enjoy full academic freedom and publish their research results freely.
- They have the opportunity to teach courses and supervise doctoral students, and have the flexibility to incorporate teaching into their research agenda.
- They are provided with outstanding technical and administrative support facilities as well as internationally competitive compensation packages.

Over the next decade, the institute will grow to a strength of about 17 tenured and tenure-track researchers, and about 100 doctoral and post-doctoral positions. Additional growth is expected through outside funding. We maintain an open, international and diverse work environment and seek applications from outstanding researchers regardless of national origin or citizenship. The working language is English; knowledge of the German language is not required for a successful career at the institute.

The institute's locations in Kaiserslautern and Saarbruecken, Germany, offer a high standard of living, numerous cultural attractions and beautiful surroundings in the center of Europe, as well as a stimulating, competitive and collaborative work environment. In immediate proximity are the MPI for Informatics, Saarland University, the Technical University Kaiserslautern, the German Center for Artificial Intelligence (DFKI), and the Fraunhofer Institutes for Experimental Software Engineering and for Industrial Mathematics.

Qualified candidates should apply online at **http://www.mpi-sws.org/application.**

The review of applications will begin on January 14, 2008, and applicants are strongly encouraged to submit applications by that date; however, applications will continue to be accepted until February 29, 2008.

The Max Planck Society is committed to increasing the representation of minorities, women and individuals with physical disabilities in Computer Science. We particularly encourage such individuals to apply.

The recently founded MPI for Software Systems joins a network of almost eighty Max Planck Institutes (MPI), Germany's premier basic research facilities. MPIs have an established record of world-class, foundational research in the fields of medicine, biology, chemistry, physics, technology and humanities. Since 1948, MPI researchers have won 17 Nobel prizes. The new MPI aspires to meet the highest standards of excellence and international recognition in its research in software systems.

MAX-PLANCK-GESELLSCHAFT

mediately. To assure consideration materials from applicants should be received by February 1, 2008. Successful applicants must demonstrate potential for research and teaching excellence in the environment of a major research university.

To be considered, please send an email to: cs-search-08@cs.drexel.edu

Please include a cover letter, CV, brief statements describing your research program and teaching philosophy, and contact information for at least four references. Electronic submissions in PDF format are strongly preferred.

Drexel University
Department of Computer Science

Drexel Universitys Department of Computer Science (www.cs.drexel.edu) invites applications for tenure-track faculty positions at all levels. The preferred interest is ARTIFICIAL INTELLIGENCE and MULTI-AGENT SYSTEMS, although exceptional applicants in other areas will be considered. The department has expanding graduate research and education programs in software engineering, graphics and vision, information assurance and security, human-computer interaction, high-performance computing and symbolic computation. We specialize in interdisciplinary and applied research and are supported by several major federal research grants from NSF, DARPA, ONR, DoD, DoE and NIST, as well as by private sources such as Intel, Nissan, NTT, and Lockheed Martin. The department offers BS, BA, MS, and Ph.D. degrees in computer science as well as BS and MS degrees in software engineering. Drexel is a designated National Security Agency (NSA) Center of Academic Excellence in Information Assurance Education. The department has over 425 undergraduate and over 100 graduate students, with annual research expenditures in excess of $4M. Several of the Computer Science faculty are recipients of NSF CAREER or Young Investigator Awards. Review of applications begins immediately. To assure consideration materials from applicants should be received by February 1, 2008. Successful applicants must demonstrate potential for research and teaching excellence in the environment of a major research university. To be considered, please send an email to: cs-search-08@cs.drexel.edu Please include a cover letter, CV, brief statements describing your research program and teaching philosophy, and contact information for at least four references. Electronic submissions in PDF format are strongly preferred

Donald Bren School of Information and Computer Sciences
Department of Informatics at the University of California, Irvine (UCI

The Department of Informatics at the University of California, Irvine (UCI) is seeking excellent candidates for a tenure-track position in Software Engineering starting in July 2008. The position is targeted at the rank of assistant professor, but exceptional candidates at all ranks will be considered. Software engineering is a particular strength of the Department, and we are looking for candidates who both broaden and deepen our vision. More information on this and other positions is available at http://www.ics.uci.edu/employment/employ_faculty.php.

Career Opportunities

Duke University
Department of Computer Science

The Department of Computer Science at Duke University invites applications and nominations for faculty positions at all levels, to begin August 2008. We are interested in strong candidates in all active research areas of computer science, both core and interdisciplinary areas, including distributed systems, computer architecture, networking, security, database systems, algorithms, artificial intelligence, machine learning, image analysis, and computer vision.

The department is committed to increasing the diversity of its faculty, and we strongly encourage applications from women and minority candidates.

A successful candidate must have a solid disciplinary foundation and demonstrate promise of outstanding scholarship in every respect, including research and teaching. Please refer to www.cs.duke.edu for information about the department.

Applications should be submitted online at www.cs.duke.edu/facsearch. A Ph.D. in computer science or related area is required. To guarantee full consideration, applications and letters of reference should be received by January 7, 2008.

Durham, Chapel Hill, and the Research Triangle of North Carolina are thriving, family-friendly communities. Duke and the many other universities in the area offer a wealth of education and employment opportunities for spouses and families.

Duke University is an affirmative action, equal opportunity employer.

Florida International University
School of Computing and Information Sciences

Applications are invited for multiple tenure-track or tenured faculty positions at the levels of Assistant, Associate, or Full Professor. A Ph.D. in Computer Science or related areas is required. Outstanding candidates are sought in areas of (1) Software and Computer Security; (2) Software and Computer Systems; (3) Bio/Medical/Health Informatics; (4) Data Mining; and (5) Human-Computer Interface (HCI). Exceptional candidates in other areas will be considered as well. Candidates with the ability to forge interdisciplinary collaborations will be favored. Candidates for senior positions must have a proven record of excellence in research funding, publications, teaching, and professional service, as well as demonstrated ability for developing and leading collaborative research projects. Outstanding candidates for the senior positions will be considered for the endowed Ryder Professorship position. Successful candidates are expected to develop a high-quality funded research program and must be committed to excellence in teaching at both graduate and undergraduate levels.

Florida International University (FIU), the state university of Florida in Miami, is ranked by the Carnegie Foundation as a comprehensive doctoral research university with high research activity. FIU offers over 200 baccalaureate, masters and doctoral degree programs in 21 colleges and schools. With over 38,000 students, it is one of the 25 largest universities in the United States, and boasts a new and accredited Law School and the newly created College of Medicine. US News & World Report has ranked FIU among the top 100 public universities, and Kiplinger's Personal Finance magazine ranked

---

Oregon State UNIVERSITY OSU

The School of Electrical Engineering and Computer Science at Oregon State University invites applications for up to three tenure-track positions in Computer Science. The School of EECS strongly encourages teamwork and collaboration within the School, and with other departments and universities. We are particularly interested in candidates who can contribute richness and depth to our Graphics/Visualization, End-User Software Engineering and Machine Learning groups. The following areas are strong possibilities for collaboration with these groups: Computer Vision; Human Computer Interaction; Natural Language Processing; Parallel and Distributed Computing (including multicore and data center computing); Programming Languages; Software Engineering; and Theoretical Computer Science (including algorithms and optimization). Applicants should have an earned doctorate in Computer Science/Computer Engineering and demonstrate a strong commitment to high-quality undergraduate and graduate teaching and the development of a vibrantresearch program. OSU is one of only two American universities to hold the Land Grant, Sea Grant, Sun Grant, and Space Grant designation and is the only Oregon institution recognized for its "very high research activity" (RU/VH) by the Carnegie Foundation for the Advancement of Teaching. With a faculty of 45,the School of EECS enrolls 1300 undergraduate and 300 MS/PhD students.For more information, including instructions for application, visit http://www.eecs.oregonstate.edu. OSU is an AA/EOE.

Chair, Department of Computer and Information Sciences

TEMPLE UNIVERSITY®

Located in Philadelphia, the 5th-largest city in the U.S., Temple University is a comprehensive Carnegie Research I Institution that serves more than 34,000 students. The CIS Department has two undergraduate degree programs, one in Computer Science and one in Information Science and Technology; a master's program in CIS; and a PhD program in CIS. It has undergone considerable growth in research in the past 7 years, during which research funding, publication rates and the number of PhD students have more than doubled.

Applicants are expected to have outstanding research accomplishments in computer and information sciences and a commitment to quality undergraduate and graduate programs and instruction. Applications from candidates with significant interdisciplinary interests are encouraged, and administrative experience at any academic level is an asset. Candidates from an industry with a strong record of research and administrative leadership are also encouraged.

Applications consisting of curriculum vitae; a statement of recent achievements, research and teaching goals; up to three representative publications; a vision statement; and names and addresses of at least three references should be submitted online at http://academicjobsonline.org.

Review of candidates will start on February 1, 2008 and will continue until the position is filled.

For further information, please visit the **Department of Computer and Information Sciences Web site at www.temple.edu/cis or e-mail to Dr. Longin Jan Latecki, Chair, Department Chair Search Committee: latecki@temple.edu.**

Temple University is an equal opportunity, equal access, affirmative action employer committed to achieving a diverse community. AA, EOE, m/f/d/v.

FIU among the best values in public higher education in the country in their 2006 survey.

The School of Computing and Information Sciences (SCIS) is a rapidly growing program of excellence at the University. The School has 31 faculty members (including seven new faculty members hired in the last three years), 1,200 students, and offers B.S., M.S., and Ph.D. degrees in Computer Science and B.S. and B.A. degrees in Information Technology. Its undergraduate program is the largest among the ten state universities in Florida and SCIS is the largest producer of Hispanic CS and IT graduates in the US. The Ph.D. enrollment in the School has doubled in the last four years with around 80 enrolled Ph.D. students. In 2006-07, the School received $2.7M in competitive research grants and leads the similar programs in the State of Florida in terms of per faculty annual research funding. In addition, the school receives an annual average of $2.2M of in-kind grants and donations from industry. Its research has been sponsored by NSF, NASA, NIH, ARO, ONR, NOAA, and other federal agencies. Several new faculty members have received the prestigious NSF CAREER AWARD, DoE CAREER AWARD, and IBM Faculty Research Awards. SCIS has broad and dynamic partnerships with industry. Its research groups include the NSF CREST Center for Emerging Technologies for Advanced Information Processing and High-Confidence Systems, the High Performance Database Research Center, the Center for Advanced Distributed Systems Engineering, the IBM Center for Autonomic and Grid Computing, and other research laboratories. The SCIS has excellent computing infrastructure and technology support. In addition, the SCIS faculty and students have access to the grid computing infrastructure with 1000 nodes under the Latin American Grid (LA Grid) Consortium (http://lagrid.fiu.edu), a first ever comprehensive international partnership, co-founded by IBM and FIU, linking institutions in the US, Latin America, and Spain for collaborative research, innovation and workforce development.

Applications, including a letter of interest, contact information, curriculum vitae, and the names of at least three references, should be sent to Chair of Recruitment Committee, School of Computing and Information Sciences, Florida International University, University Park, Miami, FL 33199. E-mail submission to recruit@cis.fiu.edu is preferred. The application review process will begin on January 15, 2008, and will continue until the positions are filled. Further information can be obtained from the School website http://www.cis.fiu.edu, or by e-mail to recruit@cis.fiu.edu.

Women and members of underrepresented groups are strongly encouraged to apply. Florida International University is a member of the State University System of Florida and is an equal opportunity/affirmative action/equal access employer.

Frostburg State University
Computer Science Department

ASSISTANT PROFESSOR OF INFORMATION SYSTEMS Frostburg State University seeks applications for a full-time tenure-track Assistant Professor of Computer Science to begin in the Fall 2008. Salary commensurate with experience. For more information, visit our website: www.frostburg.edu/hr/jobs.htm. FSU Is An EEO. Appropriate auxiliary aids and services for qualified individuals with disability will be provided upon request. Please notify in advance. http://www.frostburg.edu/

FX Palo Alto Laboratory, Inc.
Research Scientist in Document Image Analysis

FX Palo Alto Laboratory, Inc. (FXPAL) provides multimedia and collaboration technology research for Fuji Xerox Co., Ltd., a joint venture between Xerox Corporation of America and FujiFilm of Japan. We currently have an immediate opening for a Research Scientist with expertise in analysis of document images. Experience in document layout analysis, text analysis, graphics analysis, or in developing applications integrated with these types of analysis is desired. We are developing methods for extracting content from document images in English and Japanese and for using the extracted content in applications such as viewing and retrieval. The candidate should be interested in working on practical applications in a collaborative setting. Requires a Ph.D. in Computer Science or related field, strong development skills and excellent publication record. for more information about FXPAL, please see our site at www.fxpal.com. to apply send resumes to: fxpalresumes@fxpal.com and reference job code ACM/2

George Mason University
Department of Computer Science
Volgenau School of Information Technology and Engineering
Faculty Positions in Bioengineering

The Volgenau School of Information Technology and Engineering at George Mason Uni-

versity is building a program in bioengineering, including computational approaches to biology. As part of this multidisciplinary initiative, tenure track openings are available at the Assistant, Associate and Full Professor levels in the School. Background, experience, and interest will determine the departmental affiliation of successful applicants.

Within this initiative, the Department of Computer Science is seeking faculty members who can establish strong research and teaching programs in the area of computational biology, bioinformatics, and biometrics. Minimum qualifications include a Ph.D. in Computer Science, Bioinformatics, or a closely related field, demonstrated potential for excellence and productivity in research applying computational approaches to address fundamental questions in biology or medicine, and a commitment to high quality teaching. Candidates for a senior position must have a strong record of external research support.

The School has more than 100 full-time faculty members, with over 40 in the CS Department. The research interests of the CS department includes artificial intelligence, algorithms, computational biology and bioinformatics, computer graphics, computer vision, databases, data mining, image processing, security, knowledge engineering, parallel and distributed systems, performance evaluation, real-time systems, robotics, software engineering, visualization, and wireless and mobile computing. The department has several collaborative research and teaching activities in computational biology and bioinformatics with other Mason units. For more information, visit our Web site: http://cs.gmu.edu/.

For full consideration please submit application and application materials on-line at http://jobs.gmu.edu (position number F9086z). To apply online you will need a statement of professional goals including your perspective on teaching and research, a complete CV with publications, and the names of four references. The review of applications will begin immediately and will continue until the positions are filled.

George Mason University is a growing, innovative, entrepreneurial institution with national distinction in several academic fields. Enrollment is 30,000, with students studying in over 100 degree programs on four campuses in the greater Washington, DC area. Potential interactions with government agencies, industry, medical institutions, and other universities abound. GMU is anequal opportunity/affirma tive action employer that encourages diversity.

George Mason University
Department of Computer Science

The Department of Computer Science at George Mason University invites applications for a tenure-track faculty position at the rank of Assistant Professor beginning Fall 2008.

We are seeking a faculty member who can establish strong research and teaching programs in the area of computer game development. Applicants must have a research focus in an area in computer games technology — for example, in artificial intelligence, computer graphics, real-time animation, simulation and modeling, distributed and multiagent systems, or software engineering, as applied to computer games. Minimum qualifications include a Ph.D. in Computer Science or a related field, demonstrated potential for excellence and productivity in research, and a commitment to high quality teaching.

The department currently offers a graduate certificate in Computer Games Technology, and is adding a concentration in Computer Game Design to its undergraduate program. The Computer Game Design concentration is being

UCL

UCL Department of Computer Science

2 Faculty Positions

The Department of Computer Science at University College London (UCL) seeks applications for two faculty positions in the areas of networks, systems and ubiquitous computing, at the rank of Lecturer, Senior Lecturer, or Reader (the first equivalent to Assistant Professor and the latter two equivalent to Associate Professor in the US system), commensurate with qualifications. Areas of interest include network protocols, network architectures, wireless networks, mobile and ubiquitous systems, defences against network attacks, distributed systems, computer system security, and operating systems, all with an emphasis on experimental system-building.

Applicants must hold an earned PhD in Computer Science or a closely related field by the time they begin their appointment. They will be evaluated chiefly on the significance and novelty of their research to date, and their promise for leading a group in a fruitful programme of research. They must also demonstrate an enthusiasm for teaching at the graduate and undergraduate levels.

Further details about UCL CS, the posts, and how to apply may be found at:

http://www.cs.ucl.ac.uk/vacancies

All application materials must reach UCL by the 15th of January, 2008.

Questions about these positions may be directed to Professor Anthony Finkelstein, a.finkelstein@cs.ucl.ac.uk or Professor Mark Handley, m.handley@cs.ucl.ac.uk

developed in close collaboration with faculty in the College of Visual and Performing Arts at Mason. For more information on these and other programs offered by the department, visit our Web site: http://cs.gmu.edu/

The department has over 40 faculty members with wide-ranging research interests including artificial intelligence, algorithms, computer graphics, computer vision, databases, data mining, security, human computer interaction, parallel and distributed systems, real-time systems, robotics, software engineering, and wireless and mobile computing.

George Mason University is located in Fairfax, Virginia, a suburb of Washington, DC, and home to one of the highest concentrations of high-tech firms in the nation. There are excellent opportunities for interaction with government agencies and industry, including many game and "serious game" development companies. In particular, the Washington DC region is fast becoming a hub for the serious games industry. Fairfax is consistently rated as being among the best places to live in the country, and has an outstanding local public school system.

For full consideration please submit application and application materials on-line at http://jobs.gmu.edu (position number F9084z). To apply, you will need a statement of professional goals including your perspective on teaching and research, a complete C.V. with publications, and the names of four references. The review of applications will begin immediately and will continue until the position is filled. GMU is an equal opportunity/affirmative action employer. Women and minorities are strongly encouraged to apply.

Georgetown University
Department of Computer Science

The Department of Computer Science seeks a dynamic scholar/teacher for a senior faculty position within the department. It is expected that within a short time of coming to Georgetown, this new faculty member will assume the duties and responsibilities of department chair. With the inception of the department's first graduate degree program in 2007, the department chair will be instrumental in continuing the ambitious vision for computer science at Georgetown University, while also developing degree programs and elevating the stature of the department. As such, the individual selected must have an international reputation as a scholar, experience as a successful teacher, and demonstrated leadership ability. Subject to review by the University Committee on Rank and Tenure, this position will be a tenured appointment at the full Professor level. The department consists of 7 full-time faculty members, 5 adjunct faculty members, and a full-time administrative coordinator. Review of applications and nominations will be ongoing until the position is filled. Candidates from all areas and sub-disciplines of computer science and related areas are encouraged to apply. The current research interests of the faculty are in algorithms, artificial intelligence, databases/data mining, nonstandard computing, security, and software engineering. Please visit the department's website for additional information: http://www.cs.georgetown.edu/. Also direct specific questions to Brian Blake at blakeb@cs.georgetown.edu. Please send cover letter, curriculum vitae, research/teaching statements, and the names of 5 references to: Dr. M. Brian Blake, Department of Computer Science, Georgetown University, 37th and O Street, NW, 3rd Floor St. Mary's Hall, Washington, DC 20057-1232

Georgetown University is an Equal Opportunity/Affirmative Action Employer. We are committed to creating an environment that values and supports diversity, equity and inclusiveness across our campus community and encourage applications from qualified individuals who will help us achieve this mission.

Assistant and Associate Professors, Department of Computer Science

The **Department of Computer Science** at the **University of Calgary** seeks outstanding candidates for several tenure-track positions at the Assistant and Associate Professor levels. Of particular interest are applicants from information security, theory, computer games, and subject to budgetary approval information visualization or HCI. Applicants must possess a PhD in Computer Science or related discipline, and have strong potential to develop an excellent research record. Details for each position appear at: **www.cpsc.ucalgary.ca/department/employ**.

The Department is one of Canada's leaders, as evidenced by our commitment to excellence in research and teaching. It has an expansive graduate program and extensive state-of-the-art computing facilities. Further information about the Department is available at **www.cpsc.ucalgary.ca**. Calgary is a multicultural city and the fastest growing city in Canada. Located beside the natural beauty of the Rocky Mountains, Calgary enjoys a moderate climate and outstanding year-round recreational opportunities.

Interested applicants should send a CV, a concise description of their research area and program, a statement of teaching philosophy, and arrange to have at least three reference letters sent to: **Dr. Ken Barker**, Department of Computer Science, University of Calgary, Calgary, Alberta T2N 1N4, Canada, or via email to: **search@cpsc.ucalgary.ca**. Applications will be reviewed until the position is filled.

All qualified candidates are encouraged to apply; however, Canadians and permanent residents will be given priority. The University of Calgary respects, appreciates, and encourages diversity.

For more information on the University of Calgary and the city, please visit **www.ucalgary.ca/hr/careers.**

Hendrix College
Computer Science Department

Hendrix College, a central Arkansas liberal arts college, announces a 2-year, full-time, non-tenure-track position in computer science, starting Aug 2008. Applications will be reviewed upon receipt. Full information at http://ozark.hendrix.edu.

Hong Kong Baptist University
Department of Computer Science

1. Associate Professor/Assistant Professor in Computer Science (PR118/07-08)
2. Associate Professor/Assistant Professor in Information Systems (PR119/07-08)

The Department of Computer Science seeks outstanding applicants for Assistant or Associate Professor positions (depending on qualifications and experience) starting Fall 2008. The department, with 18 faculty members and 10 supporting staff, presently offers BSc, MSc, MPhil, and PhD programmes.

Duties and responsibilities include undergraduate and postgraduate teaching, teaching programme management, performing high-impact research, and contributing to professional or institutional services. Candidates are expected to collaborate with other colleagues in research and teaching in this collegial environment.

For Post 1: Applicants should have extensive knowledge and/or experience in at least one of the following areas: business informatics, computer graphics and animation, cyber laws and ethics, information security, software development, and Web technologies. Exceptional applicants in other areas of computer science, such as intelligent informatics, networking and multimedia, and pattern recognition, will also be considered. For Post 2: Applicants should have extensive knowledge and/or experience in at least one of the following areas: business informatics, cyber laws and ethics, information security, information systems development, information systems theories and practice, IT management, and Web technologies. Exceptional applicants in other areas of information systems and/or with relevant industrial experience will also be considered.

Applicants should possess a PhD degree in computer science, information systems, or a related field, and demonstrate a strong commitment to the undergraduate and postgraduate teaching in computer science and/or information systems at all levels, with track record in innovative research and high-impact publications, and evidence of ability to bid for and pursue externally funded research programmes. For Associate Professorship, evidence of academic leadership will be an advantage.

Terms of appointment: Rank and salary will be commensurate with qualifications and experience. Remuneration package includes con-

University of
Waterloo

Applications are invited for one or more David R. Cheriton Chairs in Software Systems. These are senior positions and include substantial research support and teaching reduction. Candidates with outstanding research records in software systems (very broadly defined) are encouraged to apply. Successful applicants who join the University of Waterloo are expected to be leaders in research, have an active graduate student program and contribute to the overall development of the School. A Ph.D. in Computer Science, or equivalent, is required, with evidence of excellence in teaching and research. Rank and salary will be commensurate with experience, and appointments are expected to commence during the 2008 calendar year. The Chairs are tenured positions.
With over 70 faculty members, the University of Waterloo's David R. Cheriton School of Computer Science is the largest in Canada. It enjoys an excellent reputation in pure and applied research and houses a diverse research program of international stature. Because of its recognized capabilities, the School attracts exceptionally well-qualified students at both undergraduate and graduate levels. In addition, the University has an enlightened intellectual property policy which vests rights in the inventor: this policy has encouraged the creation of many spin-off companies including iAnywhere Solutions Inc., Maplesoft Inc., Open Text Corp and Research in Motion. Please see our website for more information:

http://www.cs.uwaterloo.ca/

Applications should be sent by electronic mail to

cs-recruiting@cs.uwaterloo.ca

or by post to

Chair, Advisory Committee on Appointments
David R. Cheriton School of Computer Science
200 University Avenue West
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

An application should include a curriculum vitae, statements on teaching and research, and the names and contact information for at least three referees. Applicants should ask their referees to forward letters of reference to the address above. Applications will be considered as soon as possible after they are complete, and as long as positions are available.
The University of Waterloo encourages applications from all qualified individuals, including women, members of visible minorities, native peoples, and persons with disabilities. All qualified candidates are encouraged to apply; however, Canadian citizens and permanent residents will be given priority.

Fall 2007

tribution by the University to a retirement benefits scheme and/or a gratuity payable upon satisfactory completion of contract, annual leave, medical & dental benefits for appointee and family, accommodation and relocation allowance where appropriate. Initial appointment will be made on a fixed-term contract of two/three years commencing September 2008. Re-appointment thereafter is subject to mutual agreement.

Application Procedure: Application, together with curriculum vitae, brief statements of teaching and research interests, and copies of transcripts/testimonials should be sent to the Personnel Office, Hong Kong Baptist University, Kowloon Tong, Hong Kong Fax: (852) 3411-5001; e-mail: recruit@hkbu.edu.hk. Application forms can be downloaded from: [http://www.hkbu.edu.hk/~pers. Applicants should also send in samples of publications, preferably five best ones out of their most recent publications, and request four referees to send in confidential reference to the Personnel Office direct. Please quote PR number on the application and any subsequent correspondence.

Details of the University's Personal Information Collection Statement can be found at http://www.hkbu.edu.hk/~pers/job. The University reserves the right not to make an appointment for the posts advertised, and the appointment will be made according to the terms & conditions then applicable at the time of offer.

Closing Date: 31 March 2008 (or until the position is filled). Review of applications will begin from January 2008.

Indiana University of Pennsylvania
Computer Science Department

Computer Science Department at Indiana University of Pennsylvania -Tenure-track position at the Assistant/Associate Professor level. For details, call 724-357-7994 or http://www.iup.edu/humanresources/jobline.

IUP is an equal opportunity employer M/F/H/V.

Institute for Infocomm Research
Computer Science

Our passionate team at the Institute for Infocomm Research (I2R) has won international awards and competitions, set international standards, successfully commercialised our technologies through spinoffs or licensing and having fun too! So, come join us! We are looking for:

Research Fellows with PhD in Computer Science or related disciplines. You will be doing research in computer graphics. The current focus is on the technologies of the personal 3D entertainment systems. These technologies include modeling, rendering, animation, imaging, display, haptics and interactive techniques. The expected deliverables are publications, patents and prototype systems. You should have strong research track record, software design and programming experiences.

Iowa State University
Department of Computer Science

The Department of Computer Science at Iowa State University is seeking outstanding candidates to fill a tenure-track position, to commence in August, 2008. We are especially interested in applicants at the assistant professor level in Programming Languages and/or Software Engineering. Successful candidates will have demonstrated potential for outstanding research and instruction in computer science. A Ph.D. or equivalent in Computer Science or a closely related field is required. Our department currently consists of 27 full-time tenure-track faculty members. We offer B.S., M.S., and Ph.D. degrees in Computer Science and participate in new B.S. degrees in Software Engineering and in Bioinformatics and Computational Biology. We also participate in interdepartmental graduate programs in Bioinformatics and Computational Biology, Human-Computer Interactions, and Information Assurance. We have about 330 B.S. students, 60 M.S. students, and 110 Ph.D. students. Almost all graduate students are supported by research or teaching assistantships. We have strong research and educational programs in Algorithms and Complexity, Artificial Intelligence, Bioinformatics and Computational Biology, Databases, Data Mining, Information Assurance, Programming Languages, Multimedia Systems, Operating Systems and Networks, Robotics, and Software Engineering. Our department has over $6.5 million in active research grants. With the above interdisciplinary activities included, we contribute to active research and training grants totaling approximately $20 million. A dynamic faculty, a moderate teaching load (typically 3 courses per year with one course reduction for active researchers and possible further reductions for junior faculty), a strong graduate program, and a well-funded research program provide an excellent academic environment. In addition, cutting-edge research and education are nurtured through interdisciplinary interactions facilitated by the Laurence H. Baker Center for Bioinformatics and Biological Statistics, the Center for Computational Intelligence, Learning and Discovery, the Center for Integrative Animal Genomics, the Cyber Innovation Institute, the Infor-

mation Assurance Center, the Department of Energy Ames Laboratory, and the Virtual Reality Application Center. Iowa State University is a major land-grant university located in Ames, Iowa. It is a pleasant, small, cosmopolitan city with a population of over 50,000 (including about 27,000 students), a vibrant cultural scene, an excellent medical clinic, and a secondary school system that ranks among the best in the United States. Ames is frequently ranked among the best communities to live in North America: 20th nationally among best places to live (2002), 3rd nationally in terms of highly educated workforce for knowledge-based industry (2005), 12th nationally for its public schools (2006). Applicants should send a curriculum vita, including teaching and research statements and the names and addresses of at least three references, to: Chair of Search Committee Fax 515-294-0258 Department of Computer Science Tel 515-294-4377 Iowa State University E-mail: faculty-search@cs.iastate.edu Ames, Iowa 50011-1041Web; www.cs.iastate.edu Review of applications will begin on December 1, 2007 and will continue until the position is filled. Iowa State University is an equal opportunity employer. Women and members of underrepresented minorities are strongly encouraged to apply.

Louisiana State University
Department of Computer Science

ASSISTANT PROFESSOR (Two positions/ Tenure-track) Department of Computer Science The Department of Computer Science at Louisiana State University (http://www.csc.lsu.edu/) seeks candidates for two Assistant Professor (Tenure-track) positions. Through a targeted investment by the state, the university has chosen to establish a Center for Secure CyberSpace jointly with LaTech. The department provides excellent research opportunities for incoming faculty with the potential to join several existing funded interdisciplinary research programs along with major efforts such as the Louisiana Optical Network Initiative (LONI, http://www.loni.org). LONI, funded by a $40 M commitment from the state provides a 40 Gbps connection between new large-scale computing resources deployed at Louisiana Research Institutes. The infrastructure includes a statewide supercomputing grid of five 112-processor IBM p5-575 supercomputers, six 528-processor Dell PowerEdge servers and a 5,760 processor central server. These resources are connected by a 40 Gbps multi-lambda fiber-optic network, which in turn, is tied to the National Lambda Rail. LSU also has established the Center for Computation & Technology (www.cct.lsu.edu) to support high-performance computing research. The department has active research in the areas of cyber security and network security.

Ideal Candidates should have expertise in one or more of the fields specified below:

Internet and network security, security in sensor networks. · Cryptographic methods, threats and vulnerabilities in cyberspace (e.g., phishing, spoofing, identity thefts etc).· High Performance Computing that leverages any of these research areas.

Required Qualifications: Ph.D. in Computer Science, Electrical Engineering, Mathematics or a closely related field; distinguished record of scholarship commensurate with experience; exceptional potential for world-class research; commitment to both undergraduate and graduate education; excellent oral and written communication skills; a commitment to high quality professional service; active participation in college responsibilities.

An offer of employment is contingent on a satisfactory pre-employment background check. Salary and rank will be commensurate with qualifications and experience. Application deadline is January 22, 2007 or until a candidate is selected. For consideration, please submit, preferably in electronic form, your curriculum vitae (including e-mail address), statement of research and teaching interests, and the names and contact information for at least three references to: Prof. S. S. IyengarCo-Chair of Center for CyberSpace SecurityDepartment of Computer Science298 Coates HallLouisiana State UniversityRef: #026921 & #023602Baton Rouge, LA 70803E-mail: search1@csc.lsu.edu LSU IS AN EQUAL OPPORTUNITY/EQUAL ACCESS EMPLOYER

Louisiana State University
Department of Computer Science

ASSISTANT PROFESSOR (Two positions/ Tenure-track) Department of Computer Science The Department of Computer Science at Louisiana State University (http://www.csc.lsu.edu/) seeks candidates for two Assistant Professor (Tenure-track) positions. Through a targeted investment by the state, the university has chosen to establish a Center for Secure Cyberspace jointly with LaTech. The department provides excellent research opportunities for incoming faculty with the potential to join several existing funded interdisciplinary research programs along with major efforts such as the Louisiana Optical Network Initiative (LONI, http://www.loni.org). LONI, funded by a $40 M commitment from the state provides a 40 Gbps connection between new

large-scale computing resources deployed at Louisiana Research Institutes. The infrastructure includes a statewide supercomputing grid of five 112-processor IBM p5-575 supercomputers, six 528-processor Dell Power Edge servers and a 5,760 processor central server. These resources are connected by a 40 Gbps multi-lambda fiber-optic network, which in turn, is tied to the National Lambda Rail. LSU also has established the Center for Computation & Technology (www.cct.lsu.edu) to support high-performance computing research. The department has active research in the areas of cyber security and network security.

Ideal Candidates should have expertise in one or more of the fields specified below:

Internet and network security, security in sensor networks. · Cryptographic methods, threats and vulnerabilities in cyberspace (e.g., phishing, spoofing, identity thefts etc).· High Performance Computing that leverages any of these research areas.

Required Qualifications: Ph.D. in Computer Science, Electrical Engineering, Mathematics or a closely related field; distinguished record of scholarship commensurate with experience; exceptional potential for world-class research; commitment to both undergraduate and graduate education; excellent oral and written communication skills; a commitment to high quality professional service; active participation in college responsibilities.

An offer of employment is contingent on a satisfactory pre-employment background check. Salary and rank will be commensurate with qualifications and experience. Application deadline is January 22, 2007 or until a candidate is selected. For consideration, please submit, preferably in electronic form, your curriculum vitae (including e-mail address), statement of research and teaching interests, and the names and contact information for at least three references to: Prof. S. S. IyengarCo-Chair of Center for Cyberspace Security Department of Computer Science298 Coates Hall Louisiana State University Ref: #026921 & #023602 Baton Rouge, LA 70803E-mail: search1@csc.lsu.edu LSU IS AN EQUAL OPPORTUNITY/EQUAL ACCESS EMPLOYER

National Taiwan University
Department of Computer Science and Information Engineering

The Department of Computer Science and Information Engineering, the Graduate Institute of Networking and Multimedia, and the Graduate Institute of Biomedical Electronics and Bioinformatics, all of National Taiwan University, have faculty openings at all ranks beginning in August 2008. Highly qualified candidates in all areas of computer science/engineering and bioinformatics are invited to apply. A Ph.D. or its equivalent is required. Applicants are expected to conduct outstanding research and be committed to teaching. Candidates should send a curriculum vitae, three letters of reference, and supporting materials before February 28, 2008, to Prof Yuh-Dauh Lyuu, Department of Computer Science and Information Engineering, National Taiwan University, No 1, Sec 4, Roosevelt Rd., Taipei 106, Taiwan.

National University of Singapore (NUS)
Head, Department of Computer Science

The National University of Singapore (NUS) invites nominations and applications for the position of Head, Computer Science Department.

NUS (http://www.nus.edu.sg) has about 23500 undergraduate and 9000 graduate students from 88 countries, with close teaching-research association with 14 national-level, 16 university-level and 80 faculty-based research institutes and centres. The university's global education program has colleges in Silicon Valley, Philadelphia, Bangalore, Shanghai and Stockolm. In 2006, Newsweek's ranking of universities listed NUS as 31st globally and 3rd in Asia/Australasia.

The School of Computing is one of the 14 faculties in NUS. It has two departments: Information Systems (IS) and Computer Science (CS). The CS Department (http://www.comp.nus.edu.sg/cs/) has some 80 faculty members, many of whom regularly publish in prestigious conferences and journals, as well as serve on their program committees and editorial boards.

We seek a Head who can take the Department to the next level. The candidate should be an internationally-recognized researcher with credentials appropriate for a tenure-track appointment as Professor, and who has experience in technical leadership and team management.

The salary and benefits are internationally competitive. The preferred start date for this appointment is July 1, 2008. Review of applications will begin on receipt and continue until the position is filled.

To apply, please send a resume, statements on research, teaching and leadership, and five references to: Prof. Y.C. Tay, Head Search Committee Chair, Department of Computer Science, National University of Singapore, Singapore 117590, Republic of Singapore Attn: SoC HR Office, E-mail: cshodrec@comp.nus.edu.sg.

NEC Laboratories America, Inc.
Research Staff Positions

The Silicon Valley branch of NEC Laboratories America, Inc., a premier research facility of NEC, has openings for research staff members in the area of data management. Our research activities in this area aim at challenges arising from large-scale enterprise data management and leveraging of web intelligence for e-business applications, including:

- Web service integration and mashup for agile data management
- DB and IR integration for efficient access to heterogeneous data
- Analytics over large-scale stream data, such as click-streams, enterprise event streams, and sensor data)
- Data-centric middleware architectures and methods for ambient/ubiquitous information systems

Responsibilities include developing IP in the form of patents and scholarly publications in leading venues, as well as creating innovative technologies for NEC products. Members are encouraged to collaborate with universities.

Qualified candidates will have:

- A Ph.D. degree in Computer Science, or related field
- Profound knowledge in data management algorithms and implementations
- Expertise in databases and data management, evidenced by papers and projects

Interested applicants should email resume to recruit@nec-labs.com and reference "Cupertino-DM" in the subject line. EOE/AA/MFDV

New Mexico Institute of Mining & Technology
Computer Science Department

Assistant/Associate Professor of Computer Science New Mexico Institute of Mining & Technology seeks applicants for a tenure-track position in its Computer Science (CS) department and Information Technology (IT) program. Appointments will generally be made at the Assistant Professor level; higher-rank appointments are possible for highly qualified applicants. Candidates must have an earned Ph.D. in CS, IT, Management, Computer Engineering, or closely related field by the time of appointment and have demonstrated potential for excellence. The ability to teach graduate and undergraduate courses, conduct research in major area of CS and IT, and attract research funding is essential. We are particularly interested in applicants with experience in interdisciplinary work involving the areas of software engineering, data mining, security, operating systems, multimedia, e-commerce, complex systems, and/or Internet technology. New Mexico Tech is a scientific and technical institute with 1700 students. The CS department offers B.S., M.S., and Ph.D. degrees and currently has more than 250 students. There are excellent facilities for research and teaching. Opportunities to interact with nearby institutions include: the National Radio Astronomy Observatory, Los

Alamos, and Sandia National Laboratories. New Mexico Tech is located in the Rio Grande Valley with fabulous weather and endless outdoor recreational opportunities. Send application material, including the names and addresses of at least three references, a brief description of research interests and accomplishments, a statement of teaching philosophy, and transcripts of graduate work to: New Mexico Tech 801 Leroy Place, Brown Hall Box 126, Socorro, NM 87801. Screening will begin immediately and continue until positions are filled. For information on these positions send inquiries via email to liebrock@cs.nmt.edu Visit out web page at http:/www.nmt.edu/.

Email applications not accepted.

AAEOE

Northwestern University
Faculty in Segal Design Institute

Northwestern University seeks a creative, energetic design faculty who will help build an exciting human-centered product and service design program. Details of the position and the application procedure can be found at:

http://www.mech.northwestern.edu/hiring.

Northwestern University is an Affirmative Action, Equal Opportunity Employer. Women and individuals in underrepresented groups are encouraged to apply. Hiring is contingent upon eligibility to work in the United States.

NXP Semiconductors
Architect Media Solutions

NXP is a top 10 semiconductor company founded by Philips more than 50 years ago. We are looking for an Architect Media Solutions to join our research facility in San Jose to develop winning media processing strategies and products.

NXP Semiconductor
Complier Architect

NXP is a top 10 semiconductor company founded by Philips more than 50 years ago. We are looking for a Compiler Architect to join our research facility in San Jose working on the award-winning TriMedia™ Media Processor.

Purdue University
Computer Engineering in the School of Electrical and Computer Engineering

The School of Electrical and Computer Engineering at Purdue University invites applications for faculty positions across the breadth of computer science/engineering at all levels. The Computer Engineering Area of the school (http://engineering.purdue.edu/ECE/Research/Areas/CompEng) has nineteen faculty members who have active research programs in areas including AI, architecture, compilers, computer vision, distributed systems, embedded systems, graphics, haptics, HCI, machine learning, multimedia systems, networking, networking applications, NLP, OS, robotics, software engineering and visualization. We will consider outstanding candidates in any area of computer science/engineering, although for at least one position there is a preference for visualization and HCI. For all positions we require a PhD in computer science/engineering or a related field and a significant demonstrated research record commensurate with the level of the position applied for. Applications should consist of a cover letter, a CV, a research statement, names and contact information for at least five references, and URLs for three to five papers. Applications should be submitted online at https://engineering.purdue.edu/Engr/AboutUs/Employment/Applications. Inquiries can be sent to compengr@ecn.purdue.edu. Applications will be considered as they are received, but for full consideration should arrive by 1 February 2008. Purdue University is an equal opportunity, equal access, affirmative action employer.

Purdue University
Department of Computer Science

The Department of Computer Science at Purdue University invites applications for tenure-track positions beginning August 2008. While outstanding candidates in all areas of Computer Science will be considered, preference will be given to applicants with a demonstrable research record in operating systems, software engineering, and theory. Candidates with a multi-disciplinary focus are also encouraged to apply. Of special interest are applicants with research focus in computational science and engineering, bioinformatics, and health-care engineering. The level of the positions will depend on the candidate's experience.

The Department of Computer Science offers a stimulating and nurturing academic environment. Forty-four faculty members direct research programs in analysis of algorithms, bioinformatics, databases, distributed and parallel computing, graphics and visualization, information security, machine learning, networking, programming languages and compilers, scientific computing, and software engineering. The department has implemented a strategic plan for future growth supported by the higher administration and recently moved into a new building. Further information about the department and more detailed descriptions of the open positions are available at http://www.cs.purdue.edu. Information about the multi-disciplinary hiring effort can be found at http://www.science.purdue.edu/COALESCE/.

All applicants should hold a PhD in Computer Science, or a closely related discipline, be committed to excellence in teaching, and have demonstrated potential for excellence in research. Salary and benefits are highly competitive. Applicants are strongly encouraged to apply electronically by sending their curriculum vitae, research and teaching statements, and names and contact information of at least three references in PDF to facsearch@cs.purdue.edu. Hard copy applications can be sent to: Faculty Search Chair, Department of Computer Science, 305 N. University Street, Purdue University, West Lafayette, IN 47907. Applicants matching one search may be considered in other relevant searches when appropriate. Review of applications will begin on October 1, 2007, and will continue until the positions are filled. Purdue University is an Equal Opportunity/Equal Access/Affirmative Action employer fully committed to achieving a diverse workforce.

Rutgers University
Tenure Track Faculty Position in Computational Biology or Biomedical Informatics

The Department of Computer Science and the BioMaPS Institute for Quantitative Biology at Rutgers University invite applications for a tenure track faculty position at the junior or senior level in the Department of Computer Science. Candidates should have a strong background and experience in computational biology or biomedical informatics, including but not limited to: structural and functional genomics and proteomics, biological networks, evolutionary and systems biology, computational modeling, machine learning and applications, large scale systems data analysis, and informatics. They should be prepared to work on interdisciplinary projects making substantive Computer Science contributions.

Applicants should submit a cover letter, curriculum vitae, research summary and statement of future research goals, together with a statement of teaching experience and interests, and arrange for four letters of recommendation to be sent on their behalf. Materials should be sent as PDF files to: Chair, Hiring Committee DCS-BioMaPS, Rutgers University, Department of Computer Science, Hill Center, Busch Campus, Piscataway, NJ 08855 (email: hiringbio@cs.rutgers.edu). For more information on the Department of Computer Science, see http://www.cs.rutgers.edu, and for the BioMaPS Institute, see http://www.biomaps.rutgers.edu. To ensure proper consideration, applications should arrive by February 1, 2008.

Rutgers University is an Affirmative Action/Equal Opportunity Employer. Women and minority candidates are especially encouraged to apply.

Rutgers University

Tenure Track Faculty Position in Computational Biomedicine, Imaging and Modeling

The Rutgers University Department of Computer Science and the Center for Computational Biomedicine, Imaging and Modeling (CBIM) seeks applicants in computer graphics and related areas, for a tenure-track faculty position starting September 2008. We're particularly interested in synergy with CBIM and thus we're excited about receiving applications primarily in all areas of computer graphics, as well as related areas such as visualization, computer vision, machine learning, and human-computer interaction. Rutgers University offers an exciting and multidisciplinary research environment and encourages collaborations between Computer Science and other disciplines.

Applicants should have earned or anticipate a Ph.D. in Computer Science or a closely related field, should show evidence of exceptional research promise, potential for developing an externally funded research program, and commitment to quality advising and teaching at the graduate and undergraduate levels. Applicants should send their curriculum vitae, a research statement addressing both past work and future plans, a teaching statement and arrange for four letters of recommendation to be sent on their behalf to hiring@cs.rutgers.edu. If electronic submission is not possible, hard copies of the application materials may be sent to:

Professor Dimitris Metaxas, Hiring Chair
Computer Science Department
Rutgers University
110 Frelinghuysen Road
Piscataway, NJ 08854

Applications should be received by February 15, 2008, for full consideration.

Rutgers University is an Affirmative Action/Equal Opportunity Employer. Women and minority candidates are especially encouraged to apply.

Sandia National Laboratories

John Von Neumann Post-Doctoral Research Fellowship

The Computational Sciences, Computer Sciences and Mathematics Center and the Computer Sciences and Information Technologies Center at Sandia National Laboratories invite outstanding candidates to apply for the 2008 John von Neumann Post-Doctoral Research Fellowship in Computational Science. This prestigious fellowship is supported by the Applied Mathematics Research Program at the U.S. Department of Energy's Office of Advanced Scientific Computing and Research. The fellowship provides an exceptional opportunity for innovative research in computational mathematics and scientific computing on advanced computing architectures with application to a broad range of science and engineering applications of national importance. Applicant must have or soon receive a PhD degree in applied/computational mathematics or related computational science and computational engineering disciplines. Applicant must have less than three-years post-doctoral experience. This appointment is for a period of one year with a possible renewal for a second year and includes a highly competitive salary, moving expenses and a generous professional travel allowance.

Sandia is one of the country's largest research facilities employing 8,700 people at major facilities in Albuquerque, New Mexico and Livermore, California. Sandia maintains research programs in a variety of areas, including computational and discrete mathematics, computational physics and engineering, systems software and tools. Sandia is a world leader in large-scale parallel computer systems, algorithms, software and applications, and provides a collaborative and highly multidisciplinary environment for computational problems at extreme scales. Sandia has a state-of-the-art parallel-computing environment, including the newly deployed Red Storm machine with over 10,000 nodes in addition to numerous large-scale clusters and visualization servers. For more details about the John von Neumann Fellowship, visit our website at www.cs.sandia.gov/VN_Web_Page.

Please apply online at http://www.sandia.gov, under Employment/ Career Opportunities/Sandia internet Careers site, reference Job Requisition Number: 58851, and please submit a CV/resume, statement of research goals, and three letters of recommendation to Pavel Bochev via electronic mail at pbboche@sandia.gov. Please reference: 58851 (Von Neumann). All applications received before January 25, 2008 will be considered; the position will remain open until filled.

U.S. Citizenship Normally Required. Equal Opportunity Employer. M/F/D/V.

South Dakota School of Mines and Technology (SDSM&T)

Assistant or Associate Professor of Computer Science

The Mathematics and Computer Science Department at South Dakota School of Mines and Technology (SDSM&T) invites applicants for a tenure-track position in Computer Science, subject to availability of funding. A Ph.D. in Computer Science or a closely related field and a commitment to teaching is required. Preference will be given to individuals with an established record of excellence in teaching and research and a strong interest in Software Engineering. The successful applicant is also expected to teach a wide range of undergraduate and graduate computer science courses, advise students, guide students in projects, and participate in scholarly activity. The typical teaching load is two to three courses per semester. While teaching is the most important part of this position, research is also expected.

SDSM&T is a small science and engineering university located in the beautiful Black Hills of South Dakota. The department offers an ABET/CAC-accredited B.S. degree in Computer Science, and an M.S. degree in Computer Science. For more information regarding the department and the university, please visit http://www.sdsmt.edu/.

Applicants must apply on-line at http://sdmines.sdsmt.edu/sdsmt/employment . If you need an accommodation to the on-line application process, please contact Human Resources (605) 394-1203. Review of applications will begin January 15, 2008 and will continue until filled. Anticipated starting date will be August 2008.

SDSM&T is an EEO/AA/ADA employer & provider.

Stanford University

School of Earth Sciences

Stanford University School of Earth Sciences invites applications for a senior tenure-track faculty appointment at either the Associate or Full Professor level in the area of computational geosciences. We welcome applicants with strong skills in computational theory and practice, as well as a working familiarity with numerical methods for large-scale problems, parallelization paradigms, and modern computer systems. The successful applicant will have research experience or interests in applications to Earth and environmental problems, including but not limited to one or more of the following areas: energy, water, other natural resources, atmospheres, oceans, fluid dynamics, geodynamics, geomechanics, hazards, seismology, electromagnetics, inversion, optimization, and imaging of the earth surface and interior. Experience with large geoscience datasets is desirable. Strong interest in or experience with research collaboration and teaching across earth and environmental science disciplines is highly desirable. We expect the successful applicant to lead development and growth of the Stanford Center for Computational Earth and Environmental Science. The Center, with its shared high productivity computing resources, seeks to expand research and educational opportunities in computational geosciences. This appointment will be with one (or jointly with two) of the four departments in the School of Earth Sciences: Energy Resources Engineering, Geological and

Environmental Sciences, Geophysics, and Environmental Earth System Science (proposed). Further information about the School of Earth Sciences and this search can be found at http://pangea.stanford.edu

Stanford University is an equal opportunity employer and is committed to increasing the diversity of its faculty. It welcomes nominations of and applications from women and minority groups, as well as others who would bring additional dimensions to the universitys research, teaching and service missions. Please apply online in electronic format (.pdf only) with the following application material: cover letter, curriculum vitae, a statement outlining research and teaching experience and interests, and the names and addresses of four referees, at

http://pangea.stanford.edu/about/jobs.php

addressed to Computational Geosciences Search Committee. Applications received by January 31, 2008 will receive full consideration, though the position will remain open until the appropriate applicant is identified.

Stevens Institute of Technology
Computer Science Department

The Computer Science Department at Stevens Institute of Technology invites applications for one or more faculty positions in the following areas: machine learning (with applications that would complement our current strengths in vision/graphics, security/privacy, programming languages, and networking); secure systems and cryptography; computational systems biology; experimental software engineering; and networking. Outstanding applicants in other areas may also be considered. Hiring is likely to be at the assistant professor level, although outstanding candidates at other levels may be considered. Applicants are expected to have a Ph.D. in Computer Science or a closely related field.

Stevens Institute of Technology is a private university located in Hoboken, New Jersey. The 55-acre campus is on the Hudson river across from midtown Manhattan within a few minutes from NYC via public transportation. Hoboken is a small upscale urban city, the residence of New Jersey's governor, and the residence of choice for many professionals working in NYC. Faculty live in Hoboken, NYC, and in bucolic suburban communities in Northern New Jersey along commuter train lines to Hoboken and NYC. Stevens' location offers excellent opportunities for collaborations with nearby universities such as NYU, Princeton, Columbia, and Rutgers/DIMACS as well as industrial research

laboratories such as Bell Labs, AT&T Labs, IBM Research, Google New York, Siemens, and the Sarnoff Corporation.

Applications should be submitted electronically at http://www.cs.stevens.edu/Search. Applications should include a curriculum vitae, teaching and research statements, and contact information for three references. Candidates should ask their references to send letters directly to the search committee. Text or PDF are strongly preferred for all application materials and reference letters. Further information is provided at the web site.

Review of applications will begin on January 15, 2008. Stevens is an Affirmative Action/Equal Opportunity employer.

State University of New York at Binghamton
Department of Computer Science
The Thomas J. Watson School of Engineering and Applied Science
http://www.cs.binghamton.edu

Applications are invited for a tenure-track position at the Assistant/Associate Professor level beginning in Fall 2008. Salary and startup packages are competitive. We are especially interested in candidates with specialization in (a) Embedded Systems and Compilers or (b) Ubiquitous Computing/Information Access or (c) Information Security or (d) Areas related to systems development. Applicants must have a Ph.D. in Computer Science or a closely related discipline by the time of appointment. Strong evidence of research capabilities and commitment to teaching are essential. We offer a significantly reduced teaching load for junior tenure track faculty for at least the first three years.

Binghamton is one of the four Ph.D. granting University Centers within the SUNY system and is nationally recognized for its academic excellence. The Department has well-established Ph.D. and M.S. programs, an accredited B.S. program and is on a successful and aggressive recruitment plan. Local high-tech companies such as IBM, Lockheed-Martin, BAE and Universal Instruments provide opportunities for collaboration. Binghamton borders the scenic Finger Lakes region of New York.

Submit a resume and the names of three references to the url address:

http://binghamton.interviewexchange.com

First consideration will be given to applications that are received by March 1, 2008. Applications will be considered until the positions are filled.

Binghamton University is an equal opportunity/affirmative action employer.

Swarthmore College
Computer Science Department

Applications are invited for a tenure track assistant professor position (pending administrative approval) and for a one year leave replacement position at the assistant professor level. Both positions begin August 2008. Swarthmore College is a small, selective liberal arts college located in a suburb of Philadelphia. The Computer Science Department offers majors and minors in computer science at the undergraduate level. The teaching load is 2 courses per semester.

Applicants must have teaching experience and should be comfortable teaching a wide range of courses at the introductory and intermediate level.

For the tenure track position we will consider all areas of CS that complement our current offerings, particularly applied algorithms. Candidates should additionally have a strong commitment to involving undergraduates in their research. A Ph.D. in CS by or near the time of appointment is required. We expect to begin interviewing in late January 2008.

For the leave replacement position, all areas of CS will be considered. Depending on administrative approval, the leave replacement position could be a multi-year appointment. A Ph.D. in CS is preferred (ABD is required). We expect to begin interviewing in mid February 2008.

See http://cs.swarthmore.edu/jobs for application submission information and more details about both positions. Swarthmore College is an equal opportunity employer. Applications from women and members of minority groups are encouraged. Applications will be accepted until the positions are filled.

Tennessee Technological University
Department of Computer Science

The Computer Science Dept has an opening for a tenure-track Asst Professor, starting 8/1/2008. PhD in CS or closely related area required. Must demonstrate potential to effectively teach CS core areas and to effectively direct students in our Internet Computing MS program. All areas of research considered, but preference will be given to candidates with expertise in distributed computing. Excellent communications skills required. Must have a commitment to the continued improvement of teaching, utilizing research-based practices. Submit TTU faculty application (http://www.tntech.edu/hr/Forms/facultyapp.pdf), CV, copies of graduate transcripts (official transcripts required upon hire), 3 current reference letters to: Faculty Search, Computer Science Dept, Tennessee Technological University, P.O. Box 5101, Cookeville, TN 38505. Email: search@csc.tntech.edu. Screening begins 2/1/2008; open until filled. AA/EEO

The Chinese University of Hong Kong
Department of Systems Engineering and Engineering Management

THE CHINESE UNIVERSITY OF HONG KONG, **Department of Systems Engineering and Engineering Management** invites

applications for posts (1) to (4) in the fields of financial engineering; information systems; logistics and supply chain management; optimization and operations research or related areas.

Post (1): Professor(s) / Associate Professor(s) / Assistant Professor(s) (Ref. 07/252(255)/2). Applicants should have (i) a doctoral degree; (ii) outstanding academic record; (iii) strong commitment to excellence in both teaching and research. Appointment(s) will normally be made on contract basis for up to three years initially, leading to longer-term appointment or substantiation later subject to budget and mutual agreement. Applications will be accepted until the posts are filled.

Post (2): Professor(s) / Associate Professor(s) / Assistant Professor(s) (visiting posts) (Ref. 07/253(255)/2). Applicants should have (i) a doctoral degree; (ii) outstanding academic record; (iii) strong commitment to excellence in both teaching and research. Appointment(s) will initially be made on visiting and contract basis, renewable subject to budget and mutual agreement. Applications will be accepted until the posts are filled.

Post (3): Research Associate Professor(s) / Research Assistant Professor(s) (Ref. 07/254(255)/2). Applicants should have (i) a doctoral degree; (ii) outstanding academic record; and (iii) strong commitment to excellent research. The appointee(s) will mainly carry out research, and may also undertake some teaching duties. Appointment(s) will initially be made on contract basis for up to two years, renewable subject to budget and mutual agreement. Applications will be accepted until the posts are filled.

Post (4): Postdoctoral Fellow(s) (Ref. 07/250(255)/3). Applicants should have (i) a doctoral degree; (ii) outstanding academic record; and (iii) strong commitment to excellent research. Appointment(s) will initially be made on contract basis for one year, renewable subject to budget and mutual agreement. Applications will be accepted until the posts are filled.

Post (5): Visiting Professors / Visiting Scholars (Ref. 07/256(255)/2). Applicants should have (i) a relevant doctoral degree; and (ii) an outstanding academic record. Those with AI background and teaching experience in computational intelligence for decision making and knowledge systems are particularly welcome. Appointments will be made on contract basis for up to three months commencing May 2008 (for teaching in the summer term). Applications will be accepted until the posts are filled. Salary will be highly competitive, commensurate with qualifications and experience. The University offers a comprehensive fringe benefit package, including medical care, plus for posts (1) to (3): a contract-end gratuity for appointments of two years or longer, and housing benefits for eligible appointees. Further information about the University and the general terms of service for appointments is available at http://www.cuhk.edu.hk/personnel. The terms mentioned herein are for reference only and are subject to revision by the University. Applications including the curriculum vitae and names of at least three referees (with e-mail and postal addresses, telephone and fax numbers), together with copies of academic qualification documents, a publication list and/or selected abstracts [except post (4)], should be sent to the Chairman, Department of Systems Engineering and Engineering Management, Room 609, William M.W. Mong Engineering Building, The Chinese University of Hong Kong, Shatin, Hong Kong. (tel: (852) 2609 8313; fax: (852) 2603 5505; e-mail: recruit@se.cuhk.edu.hk). The Personal Information Collection Statement will be provided upon request. Please quote the reference number and mark 'Application - Confidential' on cover.

The College of Staten Island
Computer Science Department

The College of Staten Island invites applications for two anticipated tenure-track positions as Assistant Professor, beginning September 2008. PhD in Computer Science or a closely related area required. Go to http://www.csi.cuny.edu/ for full description of position. Review of applications will begin immediately and continue until the position is filled. Send letter of application, a curriculum vitae, a statement of teaching and research goals, and the names, addresses, and telephone numbers of three references to: Professor Deborah Sturm, Chair, Computer Science Search Committee, Department of Computer Science, Room 1N-215, College of Staten Island, 2800 Victory Blvd, Staten Island, NY 10314. EEO/AA/ADA employer.

The George Washington University
Department of Computer Science

The Department of Computer Science at the George Washington University invites applications for a tenure-track faculty position at the rank of Assistant Professor in the area of computer and network security, to begin on September 1, 2008.

The George Washington University is a private institution that prides itself on excellent research programs, a quality undergraduate and graduate experience, and low student-teacher ratio. Located in the heart of the Nation's capital, GW affords its faculty and students unique cultural, intellectual, and research opportunities.

The Department of Computer Science offers an accredited Bachelor of Science program, a Bachelor of Arts program, and graduate degree programs at the Master's and Doctoral level. The Department has 19 full-time faculty members, several affiliated and adjunct faculty members, and over 450 students at all levels. The Department has active programs in security, networks, graphics, bioinformatics, search and data mining, human computer interaction, and machine intelligence; a center of excellence in security, funded by NSF and DARPA; collaborations with the medical school in the bioinformatics and biomedical areas; and funding from various agencies. For further information please refer to http://www.cs.gwu.edu.

Basic Qualifications: Applicants must have a doctoral degree in Computer Science or a closely related field. Applicants must have potential for developing externally funded research programs, a strong commitment to quality teaching at both undergraduate and graduate levels, and excellent communication skills.

Preferred Qualifications: All areas of security and related fields will be considered, but the department is particularly interested in system security and network security.

Application Procedure: To apply, applicants should send curriculum vitae and a research summary, and should arrange for three reference letters to be sent to us. These and other relevant supporting materials should be sent to: Chair, Faculty Search Committee, Department of Computer Science / PHIL 703, The George Washington University, Washington D.C. 20052. Only complete applications will be considered. Electronic submissions are preferred, and can be sent to cssearch@gwu.edu. Review of applications will begin January 2, 2008, and will continue until the position is filled.

The George Washington University is an Equal Opportunity/Affirmative Action employer.

The Hong Kong Polytechnic University
Department of Computing

The Department invites applications for Assistant Professors in most areas of Computing, including but not limited to Software Engineering / Biometrics / Digital Entertainment / MIS and Pervasive Computing. Applicants should have a PhD degree in Computing or closely related fields, a strong commitment to excellence in teaching and research as well as a good research publication record. Initial appointment will be made on a fixed-term gratuity-bearing contract. Re-engagement thereafter is subject

to mutual agreement. Remuneration package will be highly competitive. Applicants should state their current and expected salary in the application. Please submit your application via email to hrstaff@polyu.edu.hk. Application forms can be downloaded from http://www.polyu.edu.hk/hro/job.htm. Recruitment will continue until the positions are filled. Details of the University's Personal Information Collection Statement for recruitment can be found at http://www.polyu.edu.hk/hro/jobpics.htm.

The Ohio State University
Department of Computer Science

The Department of Computer Science and Engineering (CSE), The Ohio State University, invites applications for one tenure-track position at the Assistant Professor level. The position is open to all areas of computer science and engineering, with priority consideration given to computer architecture, networking, software engineering and programming languages, and theory Women, minorities, or individuals with disabilities are especially encouraged to apply. Applicants should hold or be completing a Ph.D. in CSE or a closely related field, and have a commitment to and demonstrated record of excellence in research and teaching. The department maintains and encourages multi-disciplinary research and education activities within and outside The Ohio State University. To apply, please submit your application via the online database. The link can be found at: http://www.cse.ohio-state.edu/department/positions.shtml Review of applications will begin in January and will continue until the position is filled. The Ohio State University is an Equal Opportunity/Affirmative Action Employer

University at Buffalo, The State University of New York
Faculty Positions in Computer Science and Engineering

Celebrating its 40th anniversary this year, the CSE Department solicits applications from excellent candidates in pervasive computing and high performance computing for openings at the assistant professor level.

The CSE department has outstanding faculty and is affiliated with successful centers devoted to biometrics, bioinformatics, biomedical computing, cognitive science, document analysis and recognition, and computer security.

Candidates are expected to have a Ph.D. in Computer Science/Engineering or related field by August 2008, with an excellent publication record and potential for developing a strong funded research program.

All applications should be submitted by January 15, 2008 electronically via recruit.cse.buffalo.edu. A cover letter, curriculum vitae, and names and email addresses of at least three references are required.

The University at Buffalo is an Equal Opportunity Employer/Recruiter.

University College London (UCL)
Department of Computer Science
2 Faculty Positions

The Department of Computer Science at University College London (UCL) seeks applications for two faculty positions in the areas of networks, systems and ubiquitous computing, at the rank of Lecturer, Senior Lecturer, or Reader (the first equivalent to Assistant Professor and the latter two equivalent to Associate Professor in the US system), commensurate with qualifications. Areas of interest include network protocols, network architectures, wireless networks, mobile and ubiquitous systems, defences against network attacks, distributed systems, computer system security, and operating systems, all with an emphasis on experimental system-building.

Applicants must hold an earned PhD in Computer Science or a closely related field by the time they begin their appointment. They will be evaluated chiefly on the significance and novelty of their research to date, and their promise for leading a group in a fruitful programme of research. They must also demonstrate an enthusiasm for teaching at the graduate and undergraduate levels.

Further details about UCL CS, the posts, and how to apply may be found at: http://www.cs.ucl.ac.uk/vacancies

All application materials must reach UCL by the 15th of January, 2008.

Questions about these positions may be directed to Professor Anthony Finkelstein, a.finkelstein@cs.ucl.ac.uk or Professor Mark Handley, m.handley@cs.ucl.ac.uk

University of California Berkeley
Electrical Engineering and Computer Sciences Department

THE UNIVERSITY OF CALIFORNIA, BERKELEY invites applications for several approved tenure-track positions in University of California Berkeley

Electrical Engineering and Computer Sciences Department

THE UNIVERSITY OF CALIFORNIA, BERKELEY invites applications for several approved tenure-track positions in ELECTRICAL ENGINEERING AND COMPUTER SCIENCES at the ASSISTANT PROFESSOR level, and one approved position at the Associate or Full Professor level, beginning Fall 2008, subject to budgetary approval. We also consider possible joint appointments with other Berkeley departments.

Applicants should have (or be about to receive) a Ph.D. in Computer Science, Electrical Engineering, Computer Engineering, or a related field, evidence of ability to establish and pursue a program of high quality research, and a strong commitment to graduate/undergraduate teaching. Prioritizing candidates' overall quality and promise over subarea of specialization, we seek applicants interested in creating innovative and far-reaching solutions to important problems in electrical engineering and computer science. We also welcome applicants working in interdisciplinary areas such as computational biology, nanoelectronics, or the uses of computing in the interests of society.

Applications should include a resume, statements of research and teaching interests, selected publications, and the names of three references who will send recommendations. Review begins November 15, 2007; candidates are urged to apply by that date. The application period closes February 15, 2008, and applications received after that date will not be considered.

To apply, go to URL:

http://www.eecs.berkeley.edu/Faculty-Jobs/

If you do not have Internet access, you may mail your application materials to:

EECS Search Committee c/o Jean Richter, 253 Cory Hall, UC Berkeley, Berkeley, CA 94720-1770. Online applications are strongly encouraged.

Recommenders providing letters should submit them directly via the URL listed above by January 18, 2008. Reference letters are NOT requested directly by the department. Recommenders may view the UC Berkeley Statement of Confidentiality at http://apo.chance.berkeley.edu/evalltr.html.

University of California is an Equal 0fportunity, Affirmative Action Employer.

ELECTRICAL ENGINEERING AND COMPUTER SCIENCES at the ASSISTANT PROFESSOR level, and one approved position at the Associate or Full Professor level, beginning Fall 2008, subject to budgetary approval. We also consider possible joint appointments with other Berkeley departments.

Applicants should have (or be about to receive) a Ph.D. in Computer Science, Electrical Engineering, Computer Engineering, or a related field, evidence of ability to establish and pursue a program of high quality research, and a strong commitment to graduate/undergraduate teaching. Prioritizing candidates' overall quality and promise over sub-area of specialization, we seek applicants interested in creat-

ing innovative and far-reaching solutions to important problems in electrical engineering and computer science. We also welcome applicants working in interdisciplinary areas such as computational biology, nanoelectronics, or the uses of computing in the interests of society.

Applications should include a resume, statements of research and teaching interests, selected publications, and the names of three references who will send recommendations. Review begins November 15, 2007; candidates are urged to apply by that date. The application period closes February 15, 2008, and applications received after that date will not be considered.

To apply, go to URL:

http://www.eecs.berkeley.edu/Faculty-Jobs/

If you do not have Internet access, you may mail your application materials to:

EECS Search Committee c/o Jean Richter, 253 Cory Hall, UC Berkeley, Berkeley, CA 94720-1770. Online applications are strongly encouraged.

Recommenders providing letters should submit them directly via the URL listed above by January 18, 2008. Reference letters are NOT requested directly by the department. Recommenders may view the UC Berkeley Statement of Confidentiality at http://apo.chance.berkeley.edu/evalltr.html.

University of California is an Equal 0fportunity, Affirmative Action Employer.

University of California, Irvine (UCI)

Assistant Professor in Medical/ Health Informatics

The University of California, Irvine (UCI) is seeking excellent candidates for a tenure-track position in Medical/Health Informatics starting in July 2008. The targeted rank is that of an assistant professor, but exceptional candidates at higher ranks will be considered. The position will be in the Donald Bren School of Information and Computer Sciences (ICS) and affiliated with UCIs Institute for Clinical Translational Science (ICTS). More information on this position is available at http://www.ics.uci.edu/employment/employ_faculty.php

University of California, Irvine (UCI)

Department of Informatics

The Department of Informatics at the University of California, Irvine (UCI) is seeking excellent candidates for a tenure-track position in organizational studies of information technology starting in July 2008. The position is targeted at the rank of associate professor, but exceptional candidates at all ranks will be considered. Organizational studies of information technology are a particular strength of the Department, and we are looking for candidates who both broaden and deepen our vision. More information on this and other positions is available at http://www.ics.uci.edu/employment/employ_faculty.php.

University of Chicago

Department of Computer Science

The Department of Computer Science at the University of Chicago invites applications from exceptionally qualified candidates in all areas of Computer Science for faculty positions at the ranks of Professor, Associate Professor, Assistant Professor, and Instructor.

The University of Chicago has the highest standards for scholarship and faculty quality, and encourages collaboration across disciplines.

The Chicago metropolitan area provides a diverse and exciting environment. The local economy is vigorous, with international stature in banking, trade, commerce, manufacturing, and transportation, while the cultural scene includes diverse cultures, vibrant theater, world-renowned symphony, opera, jazz, and blues.

The University is located in Hyde Park, a pleasant Chicago neighborhood on the Lake Michigan shore.

Please send applications or nominations to:

Professor Stuart A. Kurtz, Chairman
Department of Computer Science
The University of Chicago
1100 E. 58th Street, Ryerson Hall
Chicago, IL. 60637-1581

or to:

apply-077714@mailman.cs.uchicago.edu

(attachments can be in pdf, postscript, or Word).

Complete applications consist of (a) a curriculum vitae, including a list of publications, (b) forward-looking research and teaching statements. Complete applications for Assistant Professor and Instructor positions also require (c) three letters of recommendation, sent to

recommend-077714@mailman.cs.uchicago.edu

or to the above postal address, including one that addresses teaching ability. Applicants must have completed, or will soon complete, a doctorate degree. We will begin screening applications on December 15, 2007. Screening will continue until all available positions are filled. The University of Chicago is an equal opportunity/affirmative action employer.

University of Colorado Boulder

Faculty Position in Computational Biology

The University of Colorado at Boulder invites applications for a tenure-track faculty position in the broad areas of computational biology and bioinformatics, under the auspices of the Colorado Initiative in Molecular Biotechnology (CIMB). Individuals with interests in developing and applying computational or mathematical methods to biological systems are encouraged to apply. Areas of interest may include but are not limited to: machine learning, data mining, high-performance computing, image analysis, databases, and algorithms.

CIMB is a program which integrates faculty from the departments of Applied Mathematics; Chemical & Biological Engineering; Chemistry & Biochemistry; Computer Science; Ecology and Evolutionary Biology; Integrative Physiology; Mechanical Engineering; Molecular, Cellular & Developmental Biology; and Physics (http://bayes.colorado.edu/biotech). A successful candidate may be rostered in any one of these departments. The position is at the ASSISTANT PROFESSOR level, although senior candidates at higher ranks will be considered. Candidates must have a Ph.D. degree and a demonstrated commitment to teaching at undergraduate and graduate levels, and will be expected to develop an internationally recognized research program.

Applicants should submit a curriculum vitae, statements of research and teaching interests, and arrange to have three letters of reference sent to Computational Biology Search, 347 UCB, University of Colorado, Boulder, CO 80309-0347. Application materials may be sent electronically to: CompBio@colorado.edu. Review of applications will begin on January 15, 2008 and will continue until the position is filled. The University of Colorado is sensitive to the needs of dual career couples, and is committed to diversity and equality in education and employment. See www.Colorado.edu/Arts-Sciences/Jobs/ for full job description.

University of Colorado at Boulder

Department of Computer Science

Tenure Track Faculty Position

The Department of Computer Science at the University of Colorado at Boulder seeks outstanding candidates for a tenure-track faculty position in the following areas: programming languages, software engineering, or computer and network systems. We are most interested in junior candidates but will also consider exceptional mid-career candidates.

Applications received by January 16, 2008 will be given priority consideration. The University of Colorado at Boulder is committed to diversity and equality in education and employment. We encourage applications from women and minority candidates.

For instructions on how to submit the application, please visit http://www.cs.colorado.edu/facsearch.html

Career Opportunities

University of Illinois at Springfield (UIS)
Computer Science Department

The Computer Science Department at the University of Illinois at Springfield (UIS) invites applications for a beginning assistant professor, tenure track position to begin August 2008. Please note that a Ph.D. in Computer Science or closely related field is required at the time of hire. The position involves graduate and undergraduate teaching, supervising student research, and continuing your research. Many of our classes are taught online. All areas of expertise will be considered. Review of applications will begin November 1, 2007 and continue until the position is filled or the search is terminated. Please send your vita and contact information for three references to Chair Computer Science Search Committee; One University Plaza; UHB 3100; Springfield, IL 62703-5407.

Located in the state capital, the University of Illinois at Springfield (UIS) is one of three campuses of the University of Illinois. The UIS campus serves over 4,000 students in 19 graduate and 20 undergraduate programs. The academic curriculum of the campus emphasizes a strong liberal arts core, an array of professional programs, extensive opportunities in experiential education, and a broad engagement in public affairs issues of the day. The campus offers many small classes, substantial student-faculty interaction, and a technology enhanced learning environment. Its diverse student body includes traditional, non-traditional, and international students. UIS faculty are committed teachers, active scholars, and professionals in service to society. You are encouraged to visit the university web page at http://www.uis.edu and the department's page at http://csc.uis.edu . UIS is an affirmative action/equal opportunity employer with a strong institutional commitment to recruitment and retention of a diverse and inclusive campus community. Women, minorities, veterans, and persons with disabilities are encouraged to apply.

University of Iowa
Computer Science Department
Faculty Position
Fall 2008

The Computer Science Department seeks applications for one tenure-track assistant professor position commencing August 2008. Applications from all areas of computer science and informatics are invited. We also welcome applicants doing research at the frontiers of computing in connection with other disciplines. The Department offers BA, BS, and PhD degrees in Computer Science, and in Fall 2007 added BA and BS degrees in Informatics (see http://www.cs.uiowa.edu/Informatics. Candidates must hold a PhD in computer science, informatics, or a closely related discipline. Applications received by January 15, 2008, are assured of full consideration. Applications should contain a CV, research, and teaching statements. Please have three letters of recommendation sent directly to us (pdf email preferred). Apply: Via the Web at http://www.cs.uiowa.edu/hiring/Or by email to cs_hiring@cs.uiowa.edu Or by U.S. mail to: Faculty Search Committee Computer Science Department University of Iowa 14 MacLean Hall Iowa City, IA 52242-1419 The University of Iowa is an affirmative action/equal opportunity employer. The Department and the College of Liberal Arts and Sciences are strongly committed to diversity and maintain ties to programs on campus that provide a supportive environment for women and minorities, such as the Women in Science and Engineering program. The strategic plans of the University, College, and Department reflect this commitment to diversity.

University of Kansas
Faculty Position in
Bioinformatics/Computational Biology

The Center for Bioinformatics (www.bioinformatics.ku.edu) and the Department of Electrical Engineering and Computer Science (http://www.eecs.ku.edu) at The University of Kansas invite applications for a tenure-track assistant professor position expected to begin August 18, 2008. The Bioinformatics initiative is part of a major expansion in Life Sciences and complements existing strengths in information technology, structural biology, computational chemistry, biophysics, proteomics, developmental and molecular genetics, and drug design. Duties: to establish and maintain an externally-funded research program, to participate in teaching, and to provide service. Required Qualifications: Ph.D. in a discipline related to Bioinformatics or Computer Science expected by start date of appointment; potential for excellence in research in Bioinformatics; and commitment to teaching bioinformatics and computer science courses; strong record of research accomplishments in at least one of the following areas: biomolecular networks modeling, bioinformatics databases, and computational modeling of biomolecular systems. For the full position announcement, refer to: http://www2.ku.edu/~clas/employment/FY09_Index.htm and click the Bioinformatics/EECS download button. E-mail application as a single file, including CV, letter of application, statement of past and future research and teaching interests and philosophy to: asawyer@ku.edu. Have at least three letters of reference sent separately to: Dr. Robert F. Weaver, Professor and Associate Dean, College of Liberal Arts and Sciences, c/o Anne Sawyer, The University of Kansas, 1450 Jayhawk Blvd, 200 Strong Hall, Lawrence, KS 66045-7535. Initial review of applications begins January 15, 2008 and will continue until the position is filled. EO/AA Employer.

University of Kentucky
Department of Computer Science
Assistant Professor

The University of Kentucky Computer Science Department invites applications for a tenure-track position beginning August 15, 2008 at the assistant professor level. Candidates should have a PhD in Computer Science. Review of credentials will begin on January 15, 2008, and will continue until a suitably qualified candidate is found.

We are especially interested in candidates with expertise in computer vision, image recognition, visualization, multimedia, 3-D reconstruction of environments, computer graphics, and the intersection of databases/data mining with visualization. A successful candidate will have an opportunity to be a member of The Center for Visualization & Virtual Environments, which was established in 2003 with funding from the Kentucky Office of the New Economy and has grown to include 15 associated faculty, with upwards of $17 million in grants and contracts to date.

The University of Kentucky Department of Computer Science awards B.S., M.S., and Ph.D. degrees. The Department has 23 faculty members committed to excellence in education, research and service. It has about 200 undergraduate and 150 graduate students.

The Department has strong research programs in distributed computing, computer networks, computer vision and graphics, artificial intelligence, scientific computing, cryptography, and information-based complexity.

To apply, a UK Academic Profile must be submitted at www.uky.edu/ukjobs using job #SP519818. Questions should be directed to HR/Employment (phone 1-859-257-9555 press 2 or email ukjobs@email.uky.edu), or Diane Mier (diane@cs.uky.edu) in the Computer Science Department.

We are accepting applications now. The application deadline is March 1, 2008, but may be extended as needed. Upon offer of employment, successful applicants must undergo a national background check as required by University of Kentucky Human Resources.

The University of Kentucky is an equal opportunity employer and encourages applications from minorities and women.

University of Louisiana at Lafayette
Computer Science Department
http://www.louisiana.edu/Academic/Sciences/CMPS

Applications and nominations are invited for a tenure track position at the assistant professor

level starting Fall 2008. Applicants must have a doctorate in computer science or a closely related discipline and demonstrate a commitment to excellence in undergraduate education. The successful candidate is expected primarily to teach undergraduate courses in computer science and contribute to the discipline through active research and/or other scholarly activities. He/She can also teach at the graduate level and supervise M.S. and Ph.D. students.

The Computer Science Department at the University of Louisiana at Lafayette

(UL Lafayette) is one of the first established in the nation. The department consists of 8 faculty members, and several part-time faculty and adjunct instructors. It has about 300 majors, and offers a curriculum that has been accredited by CSAB since 1987. Computing equipment specifically allocated to undergraduates includes a Unix lab of over 170 SUN workstations and a Windows NT lab of many PCs. UL Lafayette, http://www.louisiana.edu, has the distinction of having the first ACM student chapter in the country, as well as the first Louisiana chapter of the UPE honor society. The Princeton Review included UL Lafayette as one of The Best 357 Colleges in its 2005 and 2006 guides and UL Lafayette is also included in the top 20 "Cool Schools" chosen by Careers and Colleges magazine as well as The Princeton Review.

All faculty members in the department maintain an active research program. Areas of research include cognitive science, computer science education, human computer interaction, video game design, scientific visualization, software engineering, artificial intelligence, and machine learning. The department has a very close relationship with The Center for Advanced Computer Studies, http://www.cacs.louisiana.edu, which contributes to a stimulating research environment that is exceptional among undergraduate departments. The department also has a close relationship with LITE (Louisiana Immersive Technologies Enterprise) and CBIT (Center for Business & Information Technologies).

The University of Louisiana at Lafayette is located in a city with a population of about 120,000. It is the cultural and commercial center of Acadiana, which is the region settled by the French-speaking population of Acadians who were exiled from Nova Scotia in the eighteenth century.

Culturally, the region is characterized by a joie de vivre, or joy of life. Acadiana residents are hard working and fun loving. Fairs and festivals throughout the year celebrate everything from A to Z ~ alligators to zydeco music. For more than a decade, the annual Festival International de Louisiane has showcased musicians from French-speaking countries from around the world. For more information, see http://www.lafayette.org.

Applicants should send a letter of application, resume, three letters of recommendation (at least one of which should address teaching) and any other supporting material to:

Magdy A. Bayoumi, Department Head
Computer Science Department
University of Louisiana at Lafayette
P.O. Box 41771
Lafayette, LA 70504-1771

(337) 482-6768
(337) 482-5791 FAX

Applications will be reviewed until the position is filled.

The University is in compliance with Title IX of the Civil Rights Act, Section 504 of the Rehabilitation Act of 1973, and is an Equal Employment Opportunity Affirmative Action Employer.

University of Maryland, College Park

Center for Bioinformatics and Computational Biology

The University of Maryland, College Park, invites applications for faculty positions at the Assistant, Associate, or Full Professor level in the Center for Bioinformatics and Computational Biology (cbcb.umd.edu), to be appointed jointly with the Computer Science Department (www.cs.umd.edu). After hiring a new Director in 2005, the University committed the resources to recruit six additional tenured and tenure-track faculty for the Center as part of an effort to maintain a world-class research group in bioinformatics, computational biology, computer science, genetics, and genomics. Parallel searches are ongoing in the areas of evolutionary biology and human genomics.

All applicants are expected to have strong publications and research experience in the areas of biological science and computing. Senior candidates will be expected to lead internationally prominent research programs in computational aspects of genomics and bioinformatics. Experience in interdisciplinary collaboration is an important asset. Exceptional candidates from areas outside of computer science are also encouraged to apply.

The faculty will be housed in contiguous space dedicated to the Center, and will have access to a high-end computing infrastructure through the University of Maryland Institute for Advanced Computer Studies.

Applicants should apply online at
https://www.cbcb.umd.edu/hiring/online/2008.

Applications should include a cover letter, curriculum vitae, and a description of research and teaching interests. Applicants at the Assistant Professor level should provide names and contact information for at least 3 people who will provide letters of reference. Applicants for Associate or Full professor should provide the names of at least 5 references. For full consideration, applications should be received by January 3, 2008, however applications may be accepted until the position(s) are filled.

The University of Maryland is an affirmative action, equal opportunity employer. Women and minorities are encouraged to apply.

University of Massachusetts Lowell

Department of Computer Science

Tenure-track Assistant Professor

The Computer Science Department at UMass Lowell invites applications for one tenure-track assistant professor position to start in September 2008. Applicants must hold a PhD in computer science or a closely related discipline at the time of appointment. Preference will be given to outstanding candidates with demonstrated potential to develop and sustain an external funded research program in the area of computer systems security. Exceptional candidates from other major disciplines of Computer Science will also be considered.

UMass Lowell is located about 30 miles northwest of Boston in the high-tech corridor of Massachusetts. Its CS department has 18 tenured and tenure-track faculty. It offers degree programs at the bachelor's, master's, and doctoral levels.

Send current CV, statements about research and teaching, and selected publications to hiring@cs.uml.edu. In addition, have three letters of recommendations sent directly. Visit http://www.cs.uml.edu/jobopening for more information about this position.

UMass Lowell is an affirmative action, equal opportunity Title IX employer.

University of Massachusetts Amherst

Department of Computer Science

The University of Massachusetts Amherst invites applications for tenure-track faculty positions at the assistant professor level. Applicants must have a Ph.D. in Computer Science or related area and should show evidence of exceptional research promise. Candidates with an established record of strong research may also apply for positions other than at the assistant professor level. We particularly welcome candidates who would thrive in a highly collaborative environment in which projects often span several research groups. The department is committed to the development of a diverse faculty and student body, and is very supportive of junior faculty, providing both

formal and informal mentoring. We have a strong record of NSF CAREER awards and other early research funding. Strong applicants from all areas of Computer Science will be considered, especially theoretical computer science, vision, computational biology, software engineering and programming languages. One to three positions are expected.

The Department of Computer Science has 43 tenure and research track faculty and 180 Ph.D. students with broad interdisciplinary research interests. The department offers first-class research facilities. Please see http://www.cs.umass.edu for more information. To apply, please send a cover letter referencing search R30070 (tenure-track positions) with your vita, a research statement, a teaching statement and at least three letters of recommendation.

We also invite applications for Research Faculty (R30069) and Research Scientist, Postdoctoral Research Associate, and Research Fellow (R30068) positions in all areas of Computer Science. We have particular availability in information retrieval and search. Applicants should have a Ph.D. in Computer Science or related area (or an M.S. plus equivalent experience), and should show evidence of exceptional research promise. These positions are grant-funded; appointments will be contingent upon continued funding. To apply, please send a cover letter with your vita, a research statement and at least three letters of recommendation.

Electronic submission of application materials is recommended. Application materials may be submitted in pdf format to facrec@cs.umass.edu. Likewise, letters of recommendation may be submitted electronically to facrec@cs.umass.edu either in ascii text or pdf format. Hard copies of the application materials may be sent to: Search {fill in number from above}, c/o Chair of Faculty Recruiting, Department of Computer Science, University of Massachusetts, Amherst, MA 01003-9264. We will begin to review applications on November 1, 2007 and will continue until available positions are filled. Salary and rank commensurate with education and experience; comprehensive benefits package. Positions to be filled dependent upon funding. Inquiries and requests for more information can be sent to: facrec@cs.umass.edu. The University of Massachusetts is an Affirmative Action/Equal Opportunity employer. Women and members of minority groups are encouraged to apply.

University of North Carolina at Greensboro (UNCG)

Department of Computer Science

The University of North Carolina at Greensboro (UNCG) seeks applications for an open rank position in the Department of Computer Science. Our preference is for a full professor, although exceptional candidates at other levels will be considered. Preferred research areas are those that build in our existing areas of strength, which include artificial intelligence, databases and data mining, foundations of computer science, human-computer interaction, networking, and security. We are particularly interested in candidates that can pursue interdisciplinary research and applications in the natural and life sciences. Experience mentoring graduate students at all levels is a plus. UNCG is a public coeducational, doctoral-granting residential university chartered in 1891. The Department of Computer Science at UNCG was created in 2006 after having developed into a mature program within the Department of Mathematical Sciences, and currently offers an ABET-accredited B.S. degree and an M.S. degree. The department currently has 6 tenured faculty members who are all active in research, as well as lecturers and part-time faculty. For more information on the department and university, visit the Departments web page at http://www.uncg.edu/cmp UNC Greensboro is especially proud of the diversity of its student body and we seek to attract an equally diverse applicant pool for this position, including women and members of minority groups. We are an EEO/AA employer with a strong commitment to increasing faculty diversity and will respond creatively to the needs of dual-career couples. Submit curriculum vitae, research and teaching statements, and four letters of reference to: Dr. Stephen Tate, Department of Computer Science, University of North Carolina at Greensboro, Greensboro, NC 27402 (cssearch@uncg.edu). Informal inquiries are welcome. Review of applications will begin on January 15, 2008 and continue until the position is filled.

University of North Texas

Non-Tenure Track Assistant Professor of Computer Science.

The University of North Texas is seeking applications to fill a faculty position at its Dallas Campus in Computer Science and Information Technology Program. This is a non-tenure-track position, renewable for up to five years, beginning in fall 2008. Other than not participating in the University tenure award process, all other standard University of North Texas, College of Engineering, and appropriate academic department's personnel rules and procedures apply to this position. Salary is competitive and commensurate with experience. Summer employment is contingent on student demand and funding.

The assistant professor will teach on the UNT Dallas Campus and will be expected to teach courses primarily in information technology and computer science while working with community and industry groups to develop enrollment in the Bachelor's of Arts degree in Information Technology as well as the BS in Computer Science. The workload of the faculty member in this position will be teaching a minimum of three courses per semester. In addition, the candidates must be willing to develop a record of service to the academic program, the university, and the community.

The annual evaluation of the assistant professor will be conducted according to the guidelines of an appropriate academic department. Comments and observations from the Vice Provost of the Dallas Campus will be requested as part of the annual evaluation process. The individual will be housed at and provide instruction to the students at the UNT Dallas Campus, as well as by video conference to the UNT Denton campus.

Candidates must have an earned doctorate in Computer Science, Computer Engineering or a related field. Prior successful teaching experience at the university level and experience with distance learning is highly preferred. Successful experience in Information Technology, Software Engineering or other technology areas is desirable.

Review of applicants will begin immediately and continue until the position is filled.

Submit letter of application, complete curriculum vitae, and have at least three letters of reference mailed to:

Dr. Krishna Kavi, Chair, Computer Science and Engineering, University of North Texas. PO Box 311366, Denton, TX 76203,

The University of North Texas is an Equal Opportunity/Affirmative Action institution committed to diversity in its educational programs, thereby creating a welcoming environment for everyone.

University of Texas at Austin

Department of Computer Sciences

The Department of Computer Sciences of the University of Texas at Austin invites applications for tenure-track positions at all levels. Excellent candidates in all areas will be seriously considered. All tenured and tenure-track positions require a Ph.D. or equivalent degree in computer science or a related area at the time of employment. Successful candidates are expected to pursue an active research program, to teach both graduate and undergraduate courses, and to supervise graduate students.

The department is ranked among the top ten computer science departments in the country. It has 46 tenured and tenure-track faculty members across all areas of computer science. Many of these faculty participate in interdisciplinary programs and centers in the University, including those in Computational and Applied Mathematics, Computational Biology, and Neuroscience. Austin, the capital of Texas, is

located on the Colorado River, at the edge of the Texas Hill Country, and is famous for its live music and outdoor recreation. Austin is also a center for high-technology industry, including companies such as IBM, Dell, Freescale Semiconductor, Advanced Micro Devices, National Instruments, AT&T, Intel and Samsung. For more information please see the department web page: http://www.cs.utexas.edu/.

The department prefers to receive applications online, beginning November 1, 2007. To submit yours, please visit - http://recruiting.cs.utexas.edu/faculty/

If you cannot apply online, please send a curriculum vita, home page URL, description of research interests, and selected publications, and ask three referees to send letters of reference directly to: Faculty Search Committee, Department of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin, Texas 78712-0233, USA. Inquiries about your application may be directed to faculty-search@cs.utexas.edu. For full consideration of your application, please apply by January 15, 2008. Women and minority candidates are especially encouraged to apply. The University of Texas is an Equal Opportunity Employer.

University of Texas at San Antonio

Cyber Security Faculty Positions in Computer Science

The Department of Computer Science at The University of Texas at San Antonio (UTSA) invites applications for two tenure-track positions in Cyber Security at the Assistant, Associate or Professor level, starting Fall 2008. Candidates from all areas of cyber security will be considered, but preference will be given to those who complement our existing strengths. Responsibilities include research, teaching at graduate and undergraduate levels, and program development including close collaboration with the newly formed Institute for Cyber Security headed by Ravi Sandhu. Salary and start-up support packages are highly competitive. Inquiries should be addressed to ravi.sandhu@utsa.edu.

Required qualifications: Applicants for the Assistant Professor positions must have earned a Ph.D. prior to September 1, 2008, in Computer Science or in a related field specializing in cyber security and must demonstrate a strong potential for excellence in research and teaching. Senior applicants must demonstrate an established research program in cyber security and relevant teaching experience.

The Department of Computer Science currently has 25 faculty members and offers B.S., M.S., and Ph.D. degrees supporting a dynamic and growing program with over 400 undergraduate and 100 graduate students. The research activities and experimental facilities have been well-supported by various Federal research and infrastructure grants. The Institute for Cyber Security was created in June 2007 and is initially funded by a $3.5M competitive grant from the State of Texas. Its mission is to pursue world-class cyber-security research, education, commercialization and service with high impact in synergy with world-class partners. For further information on the Department see www.cs.utsa.edu and on the Institute see www.ics.utsa.edu.

UTSA is the largest public university in south Texas serving over 28,000 students. San Antonio has a population of over one million and is known for its rich Hispanic culture, historic attractions, affordable housing, and excellent medical facilities. Major computer industries are located in San Antonio and Austin, TX located 75 miles away. Nearby higher education and research institutions include UT Health Science Center and the Southwest Research Institute.

Applicants must submit a signed letter of application which identifies the level of appointment they wish to be considered for. Applications must include a complete dated curriculum vitae (including employment, peer-reviewed publications and grants in chronological order), a statement of research interests, and the names, addresses (postal and e-mail), and telephone numbers of at least three references. Applicants who are selected for interviews must be able to show proof that they are eligible and qualified to work in the United States. Screening of applications will begin on January 22, 2008, and will continue until the positions are filled (pending budget approval). The University of Texas at San Antonio is an Affirmative Action/Equal Opportunity Employer. Women, minorities, veterans, and individuals with disabilities are encouraged to apply. Applications may be faxed, emailed, or mailed to:

Chair of Cyber Security Faculty Search Committee
Department of Computer Science
The University of Texas at San Antonio
One UTSA Circle
San Antonio, TX 78249-0667
cybersec-search@cs.utsa.edu
FAX: 210-458-4437

University of Texas-Pan American (UTPA)

Department of Computer Science

The Department of Computer Science at the University of Texas-Pan American (UTPA) seeks applications for a tenure-track Assistant Professor position (F07/08-61)

Candidates must have Ph.D. in computer science or a closely related field and outstanding potential/proven record in teaching and active research. All areas of specialization will be considered. Desired areas are database, web applications, and web services.

UTPA is situated in the Lower Rio Grande Valley of South Texas, a strategic location at the center of social and economic change. With a population of over one million, the Rio Grande Valley is one of the fastest growing regions in the country. UTPA is a leading educator of Hispanic/Latino students, with enrollment expected to surpass 18,000 students by 2008.

The Computer Science department offers BSCS (ABET/CAC Accredited) and BS undergraduate degrees, MS in Computer Science and MS in Information Technology. It also jointly offers a BS degree in Computer Engineering with the Electrical Engineering department.

Salaries are competitive. The region has a very affordable cost-of-living. The position starts Fall 2008. Please send: (1) a cover letter, (2) vita, (3) statements of teaching and research interests, and (4) names and contact information of at least three references to: Dean's Office, Computer Science Search, College of Science and Engineering, The University of Texas-Pan American, 1201 W. University Drive, Edinburg, Texas 78541-2999. Email: COSEDeansoffice@utpa.edu . Review of materials will begin on December 15, 2007 and continue until the position is filled.

NOTE: UTPA is an Equal Opportunity/Affirmative Action employer. Women, racial/ethnic minorities and persons with disabilities are encouraged to apply. This position is security-sensitive as defined by the Texas Education Code §51.215(c) and Texas Government Code §411.094(a)(2). Texas law requires faculty members whose primary language is not English to demonstrate proficiency in English as determined by a satisfactory grade on the International Test of English as a Foreign Language (TOEFL)

University of Toronto

Department of Computer Science

The Department of Computer Science, University of Toronto, invites applications for a tenure stream appointment at the rank of Assistant Professor, to begin July 1, 2008. We are interested in candidates with research expertise in database systems. The University of Toronto is an international leader in computer science research and education, and the department enjoys strong interdisciplinary ties to other units within the University. Candidates should have (or be about to receive) a Ph.D. in computer science or a related field. They must demonstrate an ability to pursue innovative research at the highest level, and a strong commitment to

teaching. Salaries are competitive with our North American peers and will be determined according to the successful applicant's experience and qualifications. Toronto is a vibrant and cosmopolitan city, one of the most desirable in the world in which to work and live. It is also a major centre for advanced computer technologies; the department has strong interaction with the computer industry. To apply for this position, please visit http://recruit.cs.toronto.edu/. The review of applications will commence on December 15, 2007. To ensure full consideration applications should be received by January 31, 2008. The University of Toronto is strongly committed to diversity within its community and especially welcomes applications from visible minority group members, women, Aboriginal persons, persons with disabilities, members of sexual minority groups, and others who may contribute to the further diversification of ideas. All qualified candidates are encouraged to apply; however, Canadians and permanent residents will be given priority.

University of Toronto
Department of Computer Science

The Department of Computer Science, University of Toronto, invites applications for a tenure stream appointment at the rank of Assistant Professor, to begin July 1, 2008.We are especially interested in candidates with research expertise in theoretical computer science, computer graphics, and human-computer interaction, but we may also consider exceptional applications from candidates in other areas of computer science. Appointments at more senior ranks may be considered in exceptional cases. The University of Toronto is an international leader in computer science research and education, and the department enjoys strong interdisciplinary ties to other units within the University. Candidates should have (or be about to receive) a Ph.D. in computer science or a related field. They must demonstrate an ability to pursue innovative research at the highest level, and a strong commitment to teaching. Salaries are competitive with our North American peers and will be determined according to the successful applicants experience and qualifications. Toronto is a vibrant and cosmopolitan city, one of the most desirable in the world in which to work and live. It is also a major centre for advanced computer technologies; the department has strong interaction with the computer industry. To apply for this position, please visit http://recruit.cs.toronto.edu/ and follow the instructions. The review of applications will commence on December 15, 2007. To ensure full consideration applications should be received by January 31, 2008. The University of Toronto is strongly committed to diversity within its community and especially welcomes applications from visible minority group members, women, Aboriginal persons, persons with disabilities, members of sexual minority groups, and others who may contribute to the further diversification of ideas. All qualified candidates are encouraged to apply; however, Canadians and permanent residents will be given priority.

University of Washington
Assistant or Associate Professor in Human Computer Interaction

The Department of Technical Communication in the College of Engineering seeks a strong researcher and teacher specializing in human computer interaction to begin autumn 2008. We will consider candidates from a range of disciplinary backgrounds who use innovative methodological and theoretical approaches to study HCI and its applications to technology development and use. Applicants must have an earned doctorate by the date of appointment.

The successful candidate will join a vibrant faculty conducting interdisciplinary research in human computer interaction and the design of communication. Faculty research includes work in usability studies, information design, emerging communication technologies, international communication, and the design of communication systems. Faculty work is integrated with cooperative HCI-focused efforts at UW, including projects with Computer Science and Engineering, the Information School, and the School of Art. The faculty also routinely engage in HCI-related projects with affiliates from the region's high-tech industry.

The University of Washington is building a culturally diverse faculty and strongly encourages applications from women and minority candidates. The University is an affirmative action, equal opportunity employer. University of Washington faculty engage in teaching, research, and service.

Review of applications will begin November 15th, 2007.

Further information about the position and the department is available at http://www.uwtc.washington.edu/hci_job.php.

To Apply: Send (1) a letter of application, (2) current c.v., (3) statement of research and teaching goals, and (4) complete contact information for three references to: Dr. Mark Zachry, Chair, Search Committee, Department of Technical Communication, College of Engineering, University of Washington, 14 Loew Hall, Box 352195, Seattle, WA 98195-2195.

University of Washington
Computer Science & Engineering and Electrical Engineering
Tenure-Track and Research Faculty

The University of Washington's Department of Computer Science & Engineering and Department of Electrical Engineering have jointly formed a new UW Experimental Computer Engineering Lab (ExCEL). In support of this effort, the College of Engineering has committed to hiring several new faculty over the forthcoming years. All positions will be dual appointments in both departments (with precise percentages as appropriate for the candidate). This year, we have two open positions, and encourage exceptional candidates in computer engineering, at tenure-track Assistant Professor, Associate Professor, or Professor, or Research Assistant Professor, Research Associate Professor, or Research Professor to apply. A moderate teaching and service load allows time for quality research and close involvement with students. The CSE and EE departments are co-located on campus, enabling cross department collaborations and initiatives. The Seattle area is particularly attractive given the presence of significant industrial research laboratories, a vibrant technology-driven entrepreneurial community, and spectacular natural beauty. Information about ExCEL can be found at http://www.excel.washington.edu.

We welcome applications in all computer engineering areas including but not exclusively: atomics scale devices & nanotechnology, implantable and biologically-interfaced devices, synthetic molecular engineering, VLSI, embedded systems, sensor systems, parallel computing, network systems, and technology for the developing world. We expect candidates to have a strong commitment both to research and teaching. ExCEL is seeking individuals at all career levels, with appointments commensurate with the candidate's qualifications and experience. Applicants for both tenure-track and research positions must have earned a PhD by the date of appointment.

Please apply online at http://www.excel.washington.edu/jobs.html with a letter of application, a complete curriculum vitae, statement of research and teaching interests, and the names of at least four references. Applications received by January 31st, 2008 will be given priority consideration.

The University of Washington was awarded an Alfred P. Sloan Award for Faculty Career Flexibility in 2006. In addition, the University of Washington is a recipient of a National Science Foundation ADVANCE Institutional Transformation Award to increase the participation of woman in academic science and engineering careers. We are building a culturally diverse faculty and encourage applications from women and minority candidates.

University of Wisconsin-Milwaukee
Department of Computer Science

Faculty Recruitment in Biomedical Engineering We invites applications from outstanding candi-

dates for several faculty positions in Biomedical Engineering, Bioinformatics, and Computational Biology. Candidates should have a Ph.D. in Biomedical Engineering, Computer Science or in a closely related field. Candidates with an established research program and a strong record of extramural funding are preferred and invited to apply for senior positions. Exceptional junior candidates will also be considered. Qualified candidates should have strong commitment to research and teaching. We have established a strong record of recruiting outstanding junior faculty and in providing them with a nurturing and stimulating environment for career development. Several of our faculty, for example, has received the NSF Early CAREER Award. As a part of the growth and to complement our existing Ph.D. program, we have developed an exciting new interdisciplinary Ph.D. Program in Medical Informatics in collaboration with the College of Nursing, College of Health Sciences, School of Information Studies and School of Business Administration at our university and the Medical College of Wisconsin. Additional information about the new doctoral program can be found at http://www.medinf.uwm.edu/ Applicants should send a hard copy of a vita by post or by Fax, along with a statement of plans for research and teaching. We also request that at least three references be asked to send letters to: Faculty Recruitment Coordinator for Biomedical Engineering, Department of Electrical Engineering and Computer Science, University of Wisconsin-Milwaukee, PO Box 784, Milwaukee, WI 53201-0784; Fax:(414) 229-6958. Evaluation of applicants will begin January 15, 2008, and will continue until the position is filled. Women and minority candidates are strongly encouraged to apply. Additional information about the Computer Science Program can be found at http://www.cs.uwm.edu/ UWM is an equal opportunity institution committed to diversity.

Utah State University

Applications are invited for multiple faculty positions at the Assistant/Associate Professor levels, for employment beginning Fall 2008. Applicants must have completed a PhD in computer science by the time of appointment. The positions require demonstrated research success, a significant potential for attracting external research funding, excellence in teaching both undergraduate and graduate courses, the ability to supervise student research, and excellent communication skills. Candidates for Associate Professor must have demonstrated productivity. The department is interested in strengthening its focus in the following areas: Software Testing/Engineering, Parallel and Distributed Systems, Security, Broadening Participation in Computing, Computational Science/Biology, and Intelligent Agent Systems.

The department has 250 undergraduate majors, 70 MS students and 25 PhD students with 17 full time faculty. The BS degree is ABET accredited. USU is a Carnegie Research Doctoral extensive University of over 20,000 students.

Applications must be submitted using USU's online job-opportunity system. To access this job opportunity directly and begin the application process, visit https://jobs.usu.edu/applicants/Central?quickFind=52583.

The review of the applications will begin on January 3, 2008.

Washington University in Saint Louis

Department of Computer Science and Engineering

Washington University in Saint Louis Department of Computer Science and Engineering Faculty Positions The School of Engineering and Applied Science at Washington University has embarked on a major initiative to expand its programs and facilities. As part of this initiative, the Department of Computer Science and Engineering is seeking outstanding faculty in the broad area of digital systems and architecture, including embedded computing, advanced multi-core architectures and hybrid computing systems. We have a special interest in candidates seeking to develop multi-disciplinary collaborations with colleagues in related disciplines. On the applications side, this may include collaborations in systems biology, neural engineering and genetics. On the technology and basic science side, it may include collaborations in electrical engineering, materials and physics. Successful candidates must show exceptional promise for research leadership and have a strong commitment to high quality teaching at all levels.

Our faculty is engaged in a broad range of research activities including hybrid computing architectures, networking, computational biology, robotics, graphics, computer vision, and advanced combinatorial optimization. The department provides a supportive environment for research and the preparation of doctoral students for careers in research. Our doctoral graduates go on to positions of leadership in both academia and industry. The department values both fundamental research and systems research with the potential for high impact, and has a strong tradition of successful technology transfer. Limits on undergraduate enrollments and the universities growing popularity allow us to offer small classes and close personal attention to a diverse student body of exceptional quality. A faculty known for its collegiality provides a supportive environment for new arrivals. A progressive administration reaches out to academic couples seeking to co-locate, and promotes policies that reward research, teaching, and innovative new initiatives.

Washington University is one of the nations leading research universities and attracts top-ranked students from across the country and around the world. It is a medium-sized institution, with roughly 6,000 full-time undergraduates and 6,000 graduate students, allowing it to provide both a strong sense of community and a broad range of academic opportunities. Its six professional schools provide advanced education in engineering, medicine, social work, business, law, architecture and art. It has exceptional research strengths in the life sciences and medicine, creating unmatched opportunities for interdisciplinary collaborations for faculty in computer science and engineering. It has one of the most attractive university campuses anywhere, and is located in a lovely residential neighborhood, adjacent to one of the nation's largest urban parks, in the heart of a vibrant metropolitan area. St. Louis is a wonderful place to live, providing access to a wealth of cultural and entertainment opportunities, while being relatively free of the everyday hassles and pressures of larger cities.

Applicants should hold a doctorate in Computer Engineering, Computer Science or Electrical Engineering. Qualified applicants should submit a complete application (cover letter, curriculum vita, research statement, teaching statement, and names of at least three references) electronically to recruiting@cse.wustl.edu Other communications may be directed to Dr.

Jonathan Turner, jon.turner@wustl.edu. Applications will be considered as they are received.

Applications received after January 15, 2008 will receive limited consideration. Washington University is an equal opportunity/affirmative action employer.

Wayne State University

Department of Computer Science

Tenure-Track Faculty Position

The Department of Computer Science of Wayne State University invites applications for a tenure-track faculty position at the Assistant/Associate Professor level. Continuing our recent growth, we are seeking applicants in the areas of Software Engineering and Bioinformatics. Outstanding applicants in other areas will also be considered.

Candidates should have a Ph.D. in computer science or a related area. The successful candidate will have a strong commitment to both research and teaching, a strong publication record and potential for obtaining external research funding. Senior applicants should have strong publication and funding records.

Currently, the department has 19 faculty, 78 Ph.D. and 120 M.S. students. The Department's total annual R&D expenditures average between $2-3 million in research areas, includ-

ing bioinformatics, software engineering, systems, databases and image processing. Our junior faculty benefit from an extraordinarily supportive environment as demonstrated by their success in securing two recent NSF CAREER awards, as well as other very competitive research funding. Faculty actively collaborate with many other centers and departments, including the School of Medicine, which is the largest single-campus medical school in the country, and the Karmanos Cancer Institute, a nationally recognized comprehensive cancer center. More information about the department can be found at: http://www.cs.wayne.edu.

Wayne State University is a premier institution of higher education offering more than 350 undergraduate and graduate academic programs to more than 33,000 students in 11 schools and colleges. Wayne State ranks in the top 50 nationally among public research universities. As Michigan's only urban university, Wayne State fulfills a unique niche in providing access to a world-class education. The University offers excellent benefits and a competitive compensation package.

Submit applications online at http://jobs.wayne.edu, refer to Posting # 034690. and include a letter of intent, a statement of research and teaching interests, a CV, and contact information for at least three references. All applications received prior to January 11, 2008 will receive full consideration. However, applications will be accepted until the position is filled. Wayne State University is an equal opportunity/affirmative action employer.

Western Carolina University

Department of Mathematics and Computer Science Tenure-track Assistant Professor position in Computer Science beginning August2008. Requirements: Ph.D. in Computer Science or related field from appropriately accredited institution; excellence in teaching; commitment to research and service. Will teach undergraduate computer science courses. Screening begins January 10, 2008 and continues until the position is filled. Information at: https://jobs.wcu.edu/applicants/Central?quickFind=51037

Western Carolina University is an EO/AA employer that conducts background checks. Proper documentation of identity and employability are required at time of employment. Questions: Dr. William Kreahling, wkreahling@email.wcu.edu.

Western Kentucky University

Ogden College of Science and Engineering

Department of Computer Science

Department Head

Western Kentucky University's Computer Science Department announces a full-time position of Department Head beginning in the summer of 2008. The department offers an ABET/CAC accredited undergraduate program and a graduate program that is one of the largest in the state. For more information about the department visit http://cs.wku.edu/.

Western Kentucky University is a comprehensive university with a vision to become a "leading American university with international reach." Its main campus is located in Bowling Green, Kentucky, a growing city of 60,000+ population and extended campuses thrive in three other cities in Kentucky, a state noted for its high quality of life, modest cost of living, and increasing cultural diversity. With an enrollment of approximately 18,665 students in undergraduate and graduate programs, the University has grown 28% in the last ten years, and is poised to increase enrollment significantly by 2020.

Requirements for the department head position include a doctorate degree in computer science or a closely related area and being qualified for tenure at the full professor rank in Computer Science by having a record of effective teaching, research, and service. Experience in obtaining significant external grants is desirable as is previous related administrative experience. The appointed candidate will have demonstrated effective leadership abilities and will also show support for our college and university objectives. Salary is competitive.

For full consideration, candidates should submit their complete application by January 22, 2008. Review of applications will begin on January 22, 2008 and will continue until the position is filled. Applicants should submit a letter of application, curriculum vitae, a statement detailing the applicant's vision for the department and how this vision addresses current trends in computer science, three letters from professional references, and a copy of all graduate transcripts to:

Dr. Keith Andrew, Chair
Computer Science Head Search Committee
Western Kentucky University
1906 College Heights Blvd # 11077
Bowling Green, KY 42101-1077

Email the material to:
csheadsearch@wku.edu
Phone number is (270) 745-4357

All qualified individuals are encouraged to apply including women, minorities, persons with disabilities and disabled veterans. Western Kentucky University is an Affirmative Action/Equal Opportunity Employer.

جامعة كارنيجي ميلون في قطر
Carnegie Mellon QATAR CAMPUS
Member of Qatar Foundation

Carnegie Mellon

FROM ARTS TO TECHNOLOGY
CREATING INNOVATION WITH GLOBAL IMPACT

FACULTY POSITION - INFORMATION SYSTEMS • *Carnegie Mellon University in Qatar*

The undergraduate Information Systems Program at Carnegie Mellon University in Qatar seeks applicants for a teaching-track position beginning in Fall Semester, 2008. This is a career-oriented, renewable appointment that is responsible for and committed to developing and delivering excellent Information Systems courses. The Information Systems Program at Carnegie Mellon University in Qatar is a new undergraduate major intended to educate students to meet the increasing demand for Information Systems talent in Qatar and throughout the Gulf region. We are looking for candidates with backgrounds and interests in Information Systems, Information Technology, IS Management, Systems Development, or related area. A Ph.D. in Information Systems, Business Administration, or a closely-related field is required. Candidates with "ABD" status who can demonstrate significant progress toward the completion of their degree will be considered for conditional appointment. Ability to teach web development and database technologies will be required for initial teaching assignments. Special consideration may be given to candidates with research interests or experience in global systems development, IS/IT workforce development, technology adoption, or IT innovation. A demonstrated record of teaching excellence or relevant industrial working experience is particularly welcomed. Carnegie Mellon offers teaching-track appointments with multi-year contracts and promotions opportunities. While applications will be accepted at all levels of seniority, senior-level applicants should have demonstrated leadership within the academic community, have made a significant and positive impact on curriculum and student development, and have a distinguished teaching record.

The position offers competitive salaries, travel and housing allowances and other benefits packages, as well as attractive research and professional development support. A moderate teaching load allows time for quality teaching, research and close involvement with students, faculty and the local business and educational community. Applicants seeking a tenure-track position at a research university are not a good match for the needs of this position.

To apply, interested candidates should send (or arrange to have sent): 1) a letter of intent; 2) curriculum vitae and any supporting materials, including teaching philosophy; 3) three or more reference letters (Letters will not be requested directly by the Search Committee.) 4) official graduate academic transcripts and 5) additional supporting documentation which may include evidence of teaching effectiveness and/or representative research papers. **Send all materials to: Faculty Search Committee, Information Systems, Porter Hall 208, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA Email: infosys-program@andrew.cmu.edu, Fax 1-412-268-6938.**

For more information on the Information Systems program, see **http://qatar.cmu.edu/is/**
For more information on the Carnegie Mellon Qatar Campus, see **http://www.qatar.cmu.edu/**
Information on Qatar is available at: **http://www.experienceqatar.com/**

To ensure full consideration, complete applications, all supporting materials, and reference letters should be received no later than February 15, 2008.

Carnegie Mellon University is an affirmative action/equal opportunity employer, and we invite and encourage applications from women and minorities.

ACM's calendar policy is to list open computer science meetings that are Sponsor by ACM, sister societies, or other scientific, technical or educational tax-exempt organizations. Educational seminars, institutes, and courses are not included due to space limitations. Listings for conferences NOT Sponsor by ACM should include the title of the conference, Sponsor organization, a contact name and full address. Phone number, email address, URL and/or fax numbers are optional. Please address to: Calendar Items, CACM, 2 Penn Plaza, New York, NY 10121-0701; fax: (212) 869-0481; Email: calendar@acm.org. For Conferences and Workshops Sponsor or cosponsored by ACM, the calendar listing should be included with the initial documents submitted for approval to ACM. All requests for ACM sponsorship or cooperation should be addressed to: Conference Coordinator, ACM Headquarters, 2 Penn Plaza, New York, NY 10121-0701; (212) 626-0602; Email: SIGS@acm.org. The Technical Meeting Request Form (TMRF) for this purpose can be obtained from the Conference Coordinator. The TMRF should be submitted at least nine months in advance of the event to ensure time for processing the approval and to accommodate lead time for CACM listings.

The ACM calendar and calls can also be accessed on-line via the URL http://www.acm.org/events/coe.html. For further details please contact webmaster@acm.org

Conferences receiving ACM sponsorship/co-sponsorship or cooperation are noted in boldface.

2008

January 13-16

IUI '08: 12th International Conference on Intelligent User Interfaces Canary Islands, Spain, Sponsored: SIGCHI, SIGART, Contact: Jeffrey M Bradshaw, Phone: 850-232-4345, Email: jbradshaw@ai.uwf.edu

January 18-20

COMPUTE08: ACM Bangelore Chapter COMPUTE 2008 Banglore, India, Contact: R K Shyamasundar, Phone: 91 22 2280 4545, Email: shyam@tcs.tifr.res.in

January 22-25

The Tenth Australasian Computing Education Conference Wollongong, Australia, Contact: Simon, Email: simon@newcastle.edu.au

January 28-31

Multimedia Computing and Networking 2008 San Jose, CA, Contact: Reza Rejaie, Phone: 514-346-0200, Email: reza@uoregon.edu

January 28-31

International Conference on Bio-Inspired Systems and Signal Processing Funchal, Portugal, Contact: Joaquim B. Filipe, Phone: 351-91-983-3996, Email: jfilipe@insticc.org

January 31-February 1

ICUIMC '08: The Second International Conference on Ubiquitous Information Management and Communication Suwon, Korea, Sponsored: SIGKDD, Contact: Won Kim, Phone: 512-329-6673, Email: wonkimtx@gmail.com

February 11-12

International Conference on Web Search and Web Data Mining Palo Alto, CA, Contact: Marc A Najork, Phone: 650-693-2928, Email: najork@microsoft.com

February 11-14

1st International Conference on Ambient Media and Systems and Workshops Quebec, Canada, Contact: Roger M. Whitaker, Email: r.m.whitaker@cs.cardiff.ac.uk

February 15-17

Symposium on Interactive 3D Graphics and Games Redwood Shores, CA, Contact: Dr Morgan Mcguire, Email: morgan@cs.williams.edu

February 17-19

FPGA '08: ACM/SIGDA International Symposium on Field Programmable Gate Arrays Monterey, CA, Sponsored: SIGDA, Contact: Michael D Hutton, Phone: 408-544-8253, Email: mhutton1@gmail.com

February 19-22

India Software Engineering Conference Hyderabad, India, Contact: Gautam Shroff, Email: Gautam.shroff@tcs.com

February 20-23

ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming Salt Lake City, UT, Sponsored: SIGPLAN, Contact: Siddharta Chatterjee, Phone: 512-838-0008, Email: sc@us.ibm.com

February 25-26

ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems Monterey, CA, Contact: Menezes Noel, Phone: 503-264-8259, Email: noel.menezes@intel.com

February 25-27

Designing Interactive Systems Conference 2008 Cape Town, South Africa, Contact: Gary Marsden, Phone: 27-21-650-2666, Email: gaz@cs.uct.ac.za

March 1-5

ASPLOS '08: Architectural Support for Programming Languages and Operating Systems (co-located with Vee 2008) Seattle, WA, Sponsored: SIGOPS, SIGARCH, SIGPLAN, Contact: Susan J Eggers, Phone: 206-543-2118, Email: eggers@cs.washington.edu

March 3-5

CF '08: COMPUTING FRONTIERS CONFERENCE Ischia, Italy, Sponsored: SIGMICRO, Contact: Alex Ramirez, Email: alex.ramirez@bsc.es

March 12-15

HRI'08: INTERNATIONAL CONFERENCE ON HUMAN ROBOT INTERACTION Amsterdam, Netherlands, Sponsored: SIGCHI, SIGART, Contact: Kerstin Dautenhahn, Email: k.dautenhahn@herts.ac.uk

March 12-15

THE 39TH ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION Portland, OR, Sponsored: SIGCSE, Contact: John P Dougherty, Phone: 610-896-4993, Email: jd@cs.haverford.edu

March 13-14

INTERNATIONAL WORKSHOP ON SOFTWARE AND COMPILERS FOR EMBEDDED SYSTEMS Munich, Germany, Contact: Heiko Falk, Email: heiko.falk@udo.edu

March 13-17

THIRD INTERNATIONAL CONFERENCE ON BODY AREA NETWORKS Tempe, AZ, Contact: Sethuraman Panchanathan, Phone: 480-965-3699, Email: panch@asu.edu

March 16-20

THE 2008 ACM SYMPOSIUM ON APPLIED COMPUTING Fortaleza, Ceara Brazil, Contact: Roger L. Wainwright, Phone: 918-631-3143, Email: rogerw@utulsa.edu

March 26-28

EYE TRACKING RESEARCH AND APPLICATIONS Savannah, GA, Sponsored: SIGGRAPH, SIGCHI, Contact: Kari-Jouko Rajha, Phone: 358-3-35516952, Email: kkr@cs.uta.fi

March 31- April 2

WISEC'08: FIRST ACM CONFERENCE ON WIRELESS NETWORK SECURITY Alexandria, VA, Sponsored: SIGSAC, Contact: Virgil Gligor, Phone: 301-405-3642, Email: gligor@umd.edu

March 31- April 4

7TH ANNUAL ASPECT-ORIENTED SOFTWARE DEVELOPMENT CONFERENCE Brussels, Belgium, Contact: Theo D'Hondt, Email: tidhondt@vub.ac.be

April 4-5

CONSORTIUM FOR COMPUTING SCIENCES IN COLLEGES (CCSC) MIDSOUTH Russellville, AZ, Contact: James R Aman, Phone: 773-298-3454, Email: aman@sxu.edu

April 13-16

SPRING SIMULATION MULTICONFERENCE Ottawa, ON, Contact: Hassan Rajaei, Phone: 419-372-2002, Email: rajaei@cs.bgsu.edu

April 14-16

FLOPS08: 9TH INTERNATIONAL SYMPOSIUM ON FUNCTIONAL AND LOGIC PROGRAMMING Ise, Japan, Contact: Manuel V Hermenegildo, Phone: 34 91 336 7435, Email: herme@fi.upm.es

April 18-19

CONSORTIUM FOR COMPUTING SCIENCES IN COLLEGES (CCSC) SOUTH CENTRAL Corpus Christi, TX, Contact: James R Aman, Phone: 773-298-3454, Email aman@sxu.edu

April 21-25

WWW08: THE 17TH INTERNATIONAL WORLD WIDE WEB CONFERENCE Beijing, China, Contact: Yih-Farn Robin Chen, Phone: 973-360-8653, Email: chen@research.att.com

May 4-6

GREAT LAKES SYMPOSIUM ON VLSI 2008 Orlando, FL, Sponsored: SIGDA, Contact: Vijay Narayanan, Email: vijay@cse.psu.edu

May 5-8

FMX08: 13TH INTERNATIONAL CONFERENCE ON ANIMATION, EFFECTS, REALTIME AND CONTENT Stuttgart, Germany, Contact: Thomas Haegele, Phone: 490-714-1969-800, Email: Thomas.haegele@filmakademie.de

May 10-18

INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING Leipzig, Germany, Contact: Wilhelm Schaifer, Email: wilhlem@upb.de

May 27-29

THE INTERNATIONAL CONFERENCE ON ADVANCED VISUAL INTERFACES Naples, Italy, Contact: Stefano Ledialdi, Phone: 39-6-88-41962, Email: levialdi@di.uniroma1.it

June 11-13

IDC08: 7TH INTERNATIONAL CONFERENCE ON INTERACTIVE DESIGN AND CHILDREN Chicago, IL, Contact: Justine Cassell, Phone: 847-491-3534, Email: justine@media.mit.edu

June 15-20

JCDL '08: JOINT CONFERENCE ON DIGITAL LIBRARIES Pittsburgh, PA, Contact: Ronald Larsen, Phone: 412-624-5139, Email: rlarsen@pitt.edu

June 18-20

IEA/AIE-2008: 21ST INTERNATIONAL CONFERENCE ON INDUSTRIAL, ENGINEERING, & OTHER APPLICATIONS OF APPLIED INTELLIGENT SYSTEMS Wroclaw, Poland, Contact: Moonis Ali, Email: ma04@txstate.edu

July 20-23

INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND ALGEBRAIC COMPUTATION Linz/Hagenberg, Australia, Contact: Juan R. Sendra, Phone: 341-885-4902, Email: rafael.sendra@uah.es

June 23-26

WOSP '08:WORKSHOP ON SOFTWARE AND PERFORMANCE Princeton, NJ, Sponsored: SIGSOFT, SIGMETRICS, Contact: Alberto Avritzer, Phone: 908-615-4524, Email: beto5599@yahoo.com

July 7-11

ECOOP08: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING Paphos, Cyprus, Contact: Jan Vitek, Email: jv@cs.purdue.edu

July 20-23

ISSAC '08: INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND ALGEBRAIC COMPUTATION Linz/Hagenberg, Austria, Contact: Juan R Sendra, Phone: 341-885-4902, Email: rafael.sendra@uah.es

September 2-5

10TH INTERNATIONAL CONFERENCE ON HUMAN COMPUTER INTERACTION WITH MOBILE DEVICES AND SERVICES, Contact: Henri Hofte, Phone: 31-575-516319, Email: henri.terhoft@telin.nl

September 16-19

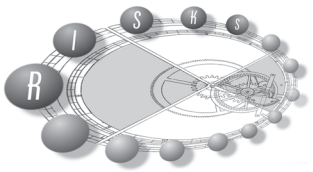
ECCE08: EUROPEAN CONFERENCE ON COGNITIVE ERGONOMICS Madeira, Portugal, Contact: Joaquim A. Jorge, Phone: 351-21-3100363, Email: jaj@inesc.pt

September 20-23

THE 10TH INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING Seoul, South Korea, Contact: Joseph McCarthy, Phone: 650-804-6987, Email: joe@interrelativity.com

December 10-13

SIGGRAPH ASIA Singapore, Sponsored: SIGGRAPH, Contact: Dr. YT Lee, Email: mvtlee@ntu.edu.sg



The Psychology of Risks

Personal risk taking is a major public-health problem in our society. It includes criminal behavior, drug addiction, compulsive gambling, accident-prone behavior, suicide attempts, and disease-promoting activities. The costs in human life, suffering, financial burden, and lost work are enormous. Some of the insights from the psychology of personal risks seem applicable to computer-related risks, and are considered here. This column is thus an orthogonal view of the past CACM "Inside Risks" columns—which have focused primarily on technological problems.

The Greeks had a word for self-defeating risk taking—*Akrasia*, which referred to incontinent behaviors that an individual performs against his or her own best interests. Clearly, there are some risks that are well considered with personal and social values. The issue that philosophers and psychologists have puzzled over has been why a person would persist in taking harmful, often impulsive risks. This question is seriously compounded when generalized to include people who are using computer systems.

Personal risk-taking behavior can arise from biological, psychological, and social causes. Computer-related risks also involve psychological and social causes—as well as economical, political, institutional, and educational causes. To understand such behavior, it must be analyzed in terms of how individuals, institutions, and the social environment perceive it and what other less-maladaptive options are available. What seems critical in assessing any such behavior is whether any control can be exerted over it, and who or what people and institutions might be aware of its consequences and able to act appropriately. Here are just a few manifestations that result from increased dependence on information technology.

Loss of a sense of community. Easy availability of excerpts from music, books, news, and other media online may lead to fewer incentives for in-person gatherings, an impersonal lack of face-to-face contact, a lessening of thoughtful feedback, and a loss of the joy of browsing among tangible entities—with many social consequences. It may also tend to reduce the general level of our intellects.

Acceleration. Instantaneous access and short-latency turnaround times as in email and instant messaging might seem to allow more time for rumination. However, the expectation of equally instantaneous responses seems to diminish the creative process and escalate the perceived needs for responses. It also seems to lead to less interest in clarity, proper grammar, and correct spelling.

Temptation. Believing that one is unobserved, anonymous, or not accountable may lead to all sorts of risks—such as clicking on untrustworthy URLs, opening up potentially dangerous attachments, and being susceptible to phishing attacks, scams, malware, and blackmail—especially when communicating with unknown people or systems. This can lead to maladaptive consequences through bad judgment and inability to recognize consequences.

Dissociation. Irrational risk behavior may arise due to problems of a modular-cognitive separation. Such behaviors are not unconsciously motivated, yet individuals and institutions are unable to connect the expression of a particular behavioral pattern with its detrimental effects. The extent to which foreseeable computer-related risks are ignored by system developers, operators, and users is quite remarkable from a psychological point of view.

Society often mythologizes artists, explorers, and scientists who take self-destructive risks as heroes who have enriched society. Often people (particularly the young) get a complicated and mixed message concerning the social value of personal risk taking. With respect to computer-related risks, modern society tends to mythologize the infallibility of computer technology and the people who develop it, or alternatively, to shoot the messenger when things go wrong rather than remediating the underlying problems.

The big difference seems to be this: In their personal lives, people tend to consciously and deliberately take risks—though often unaware of possibly serious consequences. When dealing with computer technology, people tend to take risks unconsciously and in some cases unwillingly. (On the other hand, readers of this column space are likely to be much more wary.)

In dealing with personal and computer-related risks, vigorous, compelling, and cognitively clear educational programs are essential for modulating unhealthy behavior and endorsing new attempts to deal with changing environments. C

PAUL WATSON

Leonard S. Zegans (lenz@lppi.ucsf.edu) is a psychiatrist and professor at the University of California at San Francisco Medical School.

essays, {craft, art, science} of software, python, eclipse, agile development, onward!, {generative, functional} programming, .net, open source, concurrency, smalltalk, aspects, second life, ruby, service-orientation, objects, embedded, ultra large scale {model, test}-driven passion, fun!, agents, domain-specific use cases, movies, lightning talks, systems, objective-c, development, c#, design patterns, languages, wiki, product-lines, java, refactoring, plop

DEFINE THE FUTURE OF SOFTWARE

www.oopsla.org/submit

CONFERENCE CHAIR
GAIL E. HARRIS
Instantiated Software Inc.
chair@oopsla.org

PROGRAM CHAIR
GREGOR KICZALES
University of British Columbia
papers@oopsla.org

ONWARD! CHAIR
DIRK RIEHLE
SAP Research
onward@oopsla.org

CALL FOR PAPERS
March 19, 2008
Due date for Research Program, Onward!, Development Program, Educators' Symposium, Essays and proposals for Tutorials, Panels, Workshops and DesignFest

July 2, 2008
Due date for Development Program Briefs, Doctoral Symposium and Student Volunteers

Association for Computing Machinery

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

COMMUNICATIONS

Of The Association For

COMPUTING MACHINERY

Volume 1 . Number 1 January 1958

Published by the Association for Computing Machinery