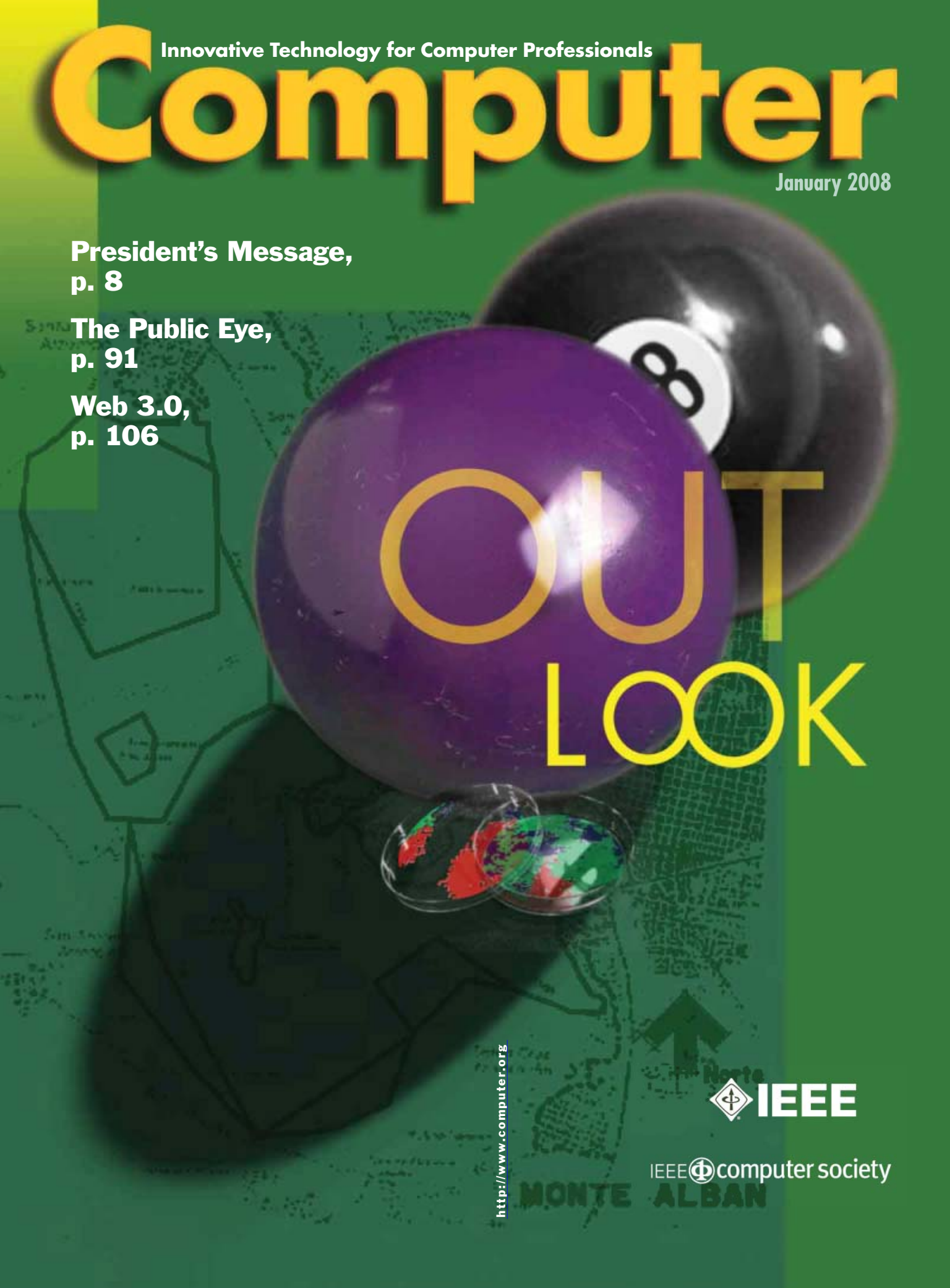
Innovative Technology for Computer Professionals

Computer

January 2008



http://www.computer.org

IEEE computer society

Innovative Technology for Computer Professionals
Computer

Circulation: *Computer* (ISSN 0018-9162) is published monthly by the IEEE Computer Society. **IEEE Headquarters**, Three Park Avenue, 17th Floor, New York, NY 10016-5997; **IEEE Computer Society Publications Office**, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; voice +1 714 821 8380; fax +1 714 821 4010; **IEEE Computer Society Headquarters**,1730 Massachusetts Ave. NW, Washington, DC 20036-1903. IEEE Computer Society membership includes $19 for a subscription to *Computer* magazine. Nonmember subscription rate available upon request. Single-copy prices: members $20.00; nonmembers $99.00.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Canadian GST #125634188. Canada Post Corporation (Canadian distribution) publications mail agreement number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8 Canada. Printed in USA.

January 2008, Volume 41, Number 1

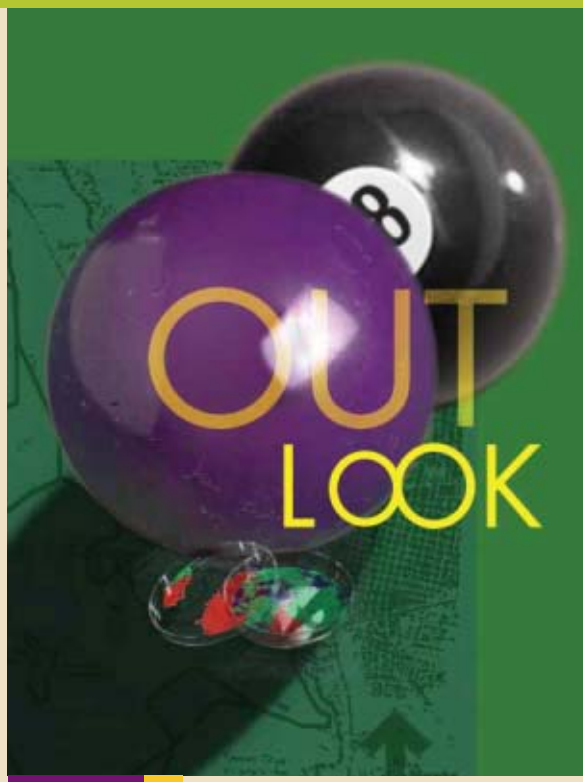
IEEE Computer Society: http://computer.org
Computer: http://computer.org/computer
computer@computer.org
IEEE Computer Society Publications Office: +1 714 821 8380

Cover design and artwork by Dirk Hagner

ABOUT THIS ISSUE

In this January Outlook issue, we continue our tradition of looking at research and technology that is likely to have an impact in the next few years. Topics covered include the emerging shift in the software industry from products to services; the dream of freeing ourselves from the straightjackets of programming and of being able to move intuitively from "played-in" scenarios to running code; and future nanoscale architectures. Other topics include a competitive pool-playing robot, evolution in the digital world, and the use of artificial intelligence and data-mining tools to study an ancient urban center.

COVER FEATURES



IEEE

IEEE computer society

The Changing Software Business: Moving from Products to Services

pp. 20-27

Michael A. Cusumano

Recently, traditional software-product sales and license fees have declined, and product-company revenues have shifted to services such as annual maintenance payments that entitle users to patches, minor upgrades, and often technical support. This shift has been especially pronounced among enterprise-software vendors.

Although online-gaming service revenues are growing fast, product sales continue to account for most game-software revenues. Platform companies like Microsoft continue to generate enormous revenues from products. But even Microsoft reported that services accounted for about 3 percent of its fiscal year 2007 revenues; just a few years ago, Microsoft derived *all* its revenues from product sales.

Can Programming Be Liberated, Period?

pp. 28-37

David Harel

The dream of being able to move intuitively from "played-in" scenarios to running code, first addressed nine years ago, remains naggingly enticing. Quite a bit of work has been carried out since then, which, while still a far cry from justifying the replacement of a dream with a plan, now seems to offer some preliminary evidence of feasibility.

The bottom line is this: There is no reason why developers shouldn't make great efforts to bring widely researched and deeply worked-out ideas in computer science to bear upon the most basic and profound activity that involves computers, namely, programming them and running the resulting programs.

An Assessment of Integrated Digital Cellular Automata Architectures

pp. 38-44

Victor Zhirnov, Ralph Cavin, Greg Leeming, and Kosmas Galatsis

As technology reaches the limits of CMOS and beyond, the physical realities of computing hardware could dictate how multicore processing evolves and what the dominant computer architecture will be.

The integration level for nanoscale electronic devices could eventually be in the range of 10^{10} to 10^{11} devices per square centimeter. At this level, long interconnects represent a significant challenge to operation, design, and manufacturing. Given these realities, future nanoscale technology could drive a migration to different information-processing and computing approaches, such as digital cellular automata.

Toward a Competitive Pool-Playing Robot

pp. 46-53

Michael Greenspan, Joseph Lam, Marc Godard, Imran Zaidi, Sam Jordan, Will Leckie, Ken Anderson, and Donna Dupuis

Since the first attempt to automate pool in the late 1980s, researchers have developed several pool-playing robotic systems as well as a training system that has a computer vision component but doesn't involve robotic actuation. Several research challenges must be addressed to advance the system further. The most difficult will emerge in competing against proficient human opponents.

Deep Green, a vision-based, intelligent robotic system to play competitive pool, currently shoots at a better-than-amateur level. Its developers seek to advance the system to successfully challenge a proficient human opponent, ultimately at a championship level.

Harnessing Digital Evolution

pp. 54-63

Philip McKinley, Betty H.C. Cheng, Charles Ofria, David Knoester, Benjamin Beckmann, and Heather Goldsby

Nearly 150 years ago, Charles Darwin explained how evolution and natural selection transformed the earliest life forms into the rich panoply of life seen today. Scientists estimate this process has been at work on Earth for at least 3.5 billion years.

In the world of computing, evolution helps humans solve complex problems in engineering and provides insight into the evolutionary process in nature. To design robust and resilient computational systems, we can take inspiration from nature. Living organisms have an amazing ability to adapt to changing environments, both in the short term through phenotypic plasticity and in the longer term through Darwinian evolution.

Mining the Social Fabric of Archaic Urban Centers with Cultural Algorithms

pp. 64-72

Robert G. Reynolds, Mostafa Ali, and Thaer Jayyousi

Applying a suite of tools from artificial intelligence and data mining to existing archaeological data from Monte Albán, a prehistoric urban center, offers the potential for building agent-based models of emergent ancient urban centers. Specifically, the authors examined the period of occupation associated with the emergence of this early site, seeking to generate a set of decision rules using data-mining techniques and then using the cultural algorithm toolkit to express the underlying social interaction between the initial inhabitants.

Future work will focus on how well the system can adjust the rules collectively to better predict terrace occupation.

IEEE computer society

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.

MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

COMPUTER SOCIETY WEB SITE: www.computer.org

OMBUDSMAN: To check membership status or report a change of address, call the IEEE Member Services toll-free number, +1 800 678 4333 (US) or +1 732 981 0060 (international). Direct all other Computer Society-related questions—magazine delivery or unresolved complaints—to help@computer.org.

CHAPTERS: Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

AVAILABLE INFORMATION: To obtain more information on any of the following, contact Customer Service at +1 714 821 8380 or +1 800 272 6657:

- Membership applications
- Publications catalog
- Draft standards and order forms
- Technical committee list
- Technical committee application
- Chapter start-up procedures
- Student scholarship information
- Volunteer leaders/staff directory
- IEEE senior member grade application (requires 10 years practice and significant performance in five of those 10)

PUBLICATIONS AND ACTIVITIES

Computer. The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Periodicals. The society publishes 14 magazines, 12 transactions, and one letters. Refer to membership application or request information as noted above.

Conference Proceedings & Books. Conference Publishing Services publishes more than 175 titles every year. CS Press publishes books in partnership with John Wiley & Sons.

Standards Working Groups. More than 150 groups produce IEEE standards used throughout the world.

Technical Committees. TCs provide professional interaction in over 45 technical areas and directly influence computer engineering conferences and publications.

Conferences/Education. The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation and certification.

Next Board Meeting: 16 May 2008, Las Vegas, NV, USA

IEEE

EXECUTIVE COMMITTEE

President: Rangachar Kasturi*
President-Elect: Susan K. (Kathy) Land*
Past President: Michael R. Williams*
VP, Electronic Products & Services: George V. Cybenko (1st VP)*
Secretary: Michel Israel (2nd VP)*
VP, Chapters Activities: Antonio Doria†
VP, Educational Activities: Stephen B. Seidman†
VP, Publications: Sorel Reisman†
VP, Standards Activities: John W. Walz†
VP, Technical & Conference Activities: Joseph R. Bumblis†
Treasurer: Donald F. Shafer*
2008–2009 IEEE Division V Director: Deborah M. Cooper†
2007–2008 IEEE Division VIII Director: Thomas W. Williams†
2008 IEEE Division VIII Director-Elect: Stephen L. Diamond†
***Computer* Editor in Chief:** Carl K. Chang†
** voting member of the Board of Governors* *† nonvoting member of the Board of Governors*

BOARD OF GOVERNORS

Term Expiring 2008: Richard H. Eckhouse, James D. Isaak, James W. Moore, Gary McGraw, Robert H. Sloan, Makoto Takizawa, Stephanie M. White
Term Expiring 2009: Van L. Eden, Robert Dupuis, Frank E. Ferrante, Roger U. Fujii, Ann Q. Gates, Juan E. Gilbert, Don F. Shafer
Term Expiring 2010: André Ivanov, Phillip A. Laplante, Itaru Mimura, Jon G. Rokne, Christina M. Schober, Ann E.K. Sobel, Jeffrey M. Voas

EXECUTIVE STAFF

Executive Director: Angela R. Burgess
Associate Executive Director: Anne Marie Kelly
Associate Publisher: Dick J. Price
Director, Administration: Violet S. Doan
Director, Finance & Accounting: John Miller

COMPUTER SOCIETY OFFICES

Washington Office. 1828 L St. N.W., Suite 1202, Washington, D.C. 20036-5104
Phone: +1 202 371 0101 • Fax: +1 202 728 9614
Email: hq.ofc@computer.org

Los Alamitos Office. 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314
Phone: +1 714 821 8380
Email: help@computer.org
Membership & Publication Orders:
Phone: +1 800 272 6657 • Fax: +1 714 821 4641
Email: help@computer.org

Asia/Pacific Office. Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • Fax: +81 3 3408 3553
Email: tokyo.ofc@computer.org

IEEE OFFICERS

President: Lewis M. Terman
President-Elect: John R. Vig
Past President: Leah H. Jamieson
Executive Director & COO: Jeffry W. Raynes
Secretary: Barry L. Shoop
Treasurer: David G. Green
VP, Educational Activities: Evangelia Micheli-Tzanakou
VP, Publication Services & Products: John Baillieul
VP, Membership & Geographic Activities: Joseph V. Lillie
VP, Standards Association Board of Governors: George W. Arnold
VP, Technical Activities: J. Roberto B. deMarca
IEEE Division V Director: Deboarh M. Cooper
IEEE Division VIII Director: Thomas W. Williams
President, IEEE-USA: Russell J. Lefevre

revised 11 Dec. 2007

JANUARY 1976

A QUARTER CENTURY (p. 6). "1976: the bicentennial year in the history of the United States, and the 25th anniversary of the founding of the IEEE Computer Society.

"It is really difficult to comprehend what has happened in our technology and in the world in this brief quarter century. Twenty-five years ago there were only a few computers—Eniac, Edvac, Univac 1, Johniac, SEAC, the IBM 604, to name most of those then in existence. Computers were only known to a few, and their promise was barely suspected. Today the annual revenue of the computer industry is measured in tens of billions. …"

[T.H. Bonn, "Reflections on the 25th Anniversary of the IEEE Computer Society," pp. 6-7.]

WOMEN DP PROFESSIONALS (p. 10). "The 1976 National Computer Conference, to be held June 7-10 in New York City, is seeking increased participation by women as session chairmen, authors, panelists, and speakers."

"According to Anita Cochrane of Bell Laboratories, a member of the '76 Program Committee, 'There are many women who make important contributions to our industry, not just a few "exceptions". To date we have contacted over 700 women in various disciplines, inviting them to participate in this conference. The response has been very heartening.'"

[Update, "1976 NCC Seeks Increased Participation of Women DP Professionals," p .10.]

MICROCOMPUTER SOFTWARE (p. 17). "A new dimension of controversy about microcomputer software was pointed out at the workshop. One view expressed was that programming a microcomputer was something that a designer could pick up without any formal training in software, and that most software tools and techniques are not really needed in most microcomputer applications. Exactly the opposite view was presented in a later session: 'Anyone who writes a program is a programmer and should be aware of and adhere to good programming practice!'"

[F.F. Coury, Guest Editor's Introduction: "Advanced Architecture and Applications of Microcomputers," pp. 16-18.]

MULTIMICROPROCESSORS (p. 30). "There are many application areas which lend themselves particularly well to multiple subsystem implementation, and hence to multimicroprocessor implementation. The commonality of these applications is the fact that it is not necessary to add an entire new working set for each new processor. Interactive systems which provide just one language (e.g., Basic or APL) need keep only one copy of the interpreter in memory, so only the user's space needs to be replicated. Also, the large class of applications where all users are making inquiries into a common data base (e.g., airline reservations, insurance company inquiries) provides an environment conducive to multimicroprocessor implementation. …"

[B.R. Borgerson, "The Viability of Multimicroprocessor Systems," pp. 28-30.]

MICROPROCESSOR SOFTWARE (p. 36). "For computers of all sizes, software has become a critical element in systems design and implementation. As dramatically as the costs of hardware have dropped, the labor-intensive costs of software have risen. In addition, there are no widely accepted and practiced techniques for controlling and estimating software development and manufacturing (size) costs, or forecasting software capability and reliability. To be sure, these problems are being addressed by software engineering, but today that is still a very primitive discipline in comparison with other computer engineering fields."

[T. Opdendyk, "Software Considerations for Microprocessors," pp. 36-38.]

STRUCTURAL MICROPROGRAMMING (p. 58). "Our working approach to block structured architectures is the following: A computer structure can be seen as level organized; the level boundary and the features contained in each level are defined following functional criteria and not following a predefined structure. The level '0' is realized by the bare machine, the first level is implemented by horizontal microprogramming, and the following level by software of increasing complexity. The simplest level of this software can be the one called firmware."

[G.F. Casaglia, "Special Feature: Nanoprogramming vs. Microprogramming," pp. 54-58.]

AIRPORT OPERATIONS (p. 67). "An airport information system designed to increase operating efficiency has been installed at the Turin, Italy, airport. Ranking third among Italian airports in volume of freight shipped, 'Citta di Torino' airport had been selected to pioneer the new ALBA (Aircraft Landing and Balancing Automation) system which uses a Hewlett-Packard 2100 minicomputer and software developed by the Aeritalia S.A.S. Group.

"ALBA is specifically designed to speed up all activities relating to flight departures at medium-sized airports with up to 100 departures a day. The system automates four functions that previously were handled manually: aircraft weight and balance calculations, management of outgoing freight and mail, passenger check-in, and passenger boarding. …"

[New Applications, "Minicomputer Streamlines Operations at Turin Airport," p. 67.]

PDFs of the articles and departments from the 1992 January issue of Computer *are available through the Computer Society's website: www.computer.org/computer.*

JANUARY 1992

SYSTEM DEVELOPMENT (p. 8). "The two main aspects of [recent] developments have to do with a carefully wrought 'vanilla' approach to system modeling and the emergence of powerful methods to execute and analyze the resulting models. It can be argued that the combined effect of these and other ideas is already showing positive signs and appears to have the potential to provide a truly major improvement in our present abilities—profoundly affecting the essence of the problem. … It will surely be a long time before reliable software for the likes of the SDI [Strategic Defense Initiative] project can be built. Such a system remains an order of magnitude too large and too critical to construct today, mainly because of its first-time-must-work nature. But I also believe that we are on the royal (main) road …"

THREE DIMENSIONS (p. 25). "A large class of problems share a common three-dimensional numerical structure and require numerous calculations on 3D vectors. …

"The 3DP, for 3-Dimensional Processor, is a parallel-computing architecture that targets these problems. It includes a hardware and software design that gives users an intuitive 3D object-oriented programming environment. It uses a C++ optimizing compiler to create assembly instructions that exploit underlying hardware capabilities for parallel processing.

"The 3DP architecture differs from traditional scalar architectures in that it operates directly on vectors. It differs from general parallel architectures in that it can solve problems that predict the behavior of highly coupled systems, and it differs from vector architectures in that it runs efficiently on length-3 vectors."

AUTHENTICATION (p. 39). "A distributed system—a collection of hosts interconnected by a network—poses some intricate security problems. A fundamental concern is authentication of local and remote entities in the system. In a distributed system, the hosts communicate by sending and receiving messages over the network. Various resources (like files and printers) distributed among the hosts are shared across the network in the form of network services provided by servers. Individual processes (clients) that desire access to resources direct service requests to appropriate servers. Aside from such client-server computing, there are many other reasons for having a distributed system. For example, a task can be divided into subtasks that are executed concurrently on different hosts.

"A distributed system is susceptible to a variety of threats mounted by intruders as well as legitimate users of the system. Indeed, legitimate users are more powerful adversaries, since they possess internal state information not usually available to an intruder (except after successful penetration of a host). …"

LOGIC PROGRAMMING (p. 83). "Algorithmic ATPG [Automatic Test Pattern Generation] for combinational circuits is an active area of research in test generation for digital systems. Application of parallel processors to this problem has shown some promising results, but much work remains. …

"Search-space partitioning shows the most promise for scalability to large numbers of processors. However, it does not answer the problem of large circuit databases created by increasing VLSI circuit sizes. Also, this technique is applicable only to hard-to-detect faults and does not address acceleration of the ATPG problem for easy-to-detect faults, which constitute the majority of the fault list for most practical circuits. Clearly, a combination of techniques or an altogether new ATPG algorithm designed for parallel processors will have to be developed to utilize the massively parallel machines with hundreds or thousands of processors that will be available in future."

HIGH-PERFORMANCE COMPUTING (p. 87). "In the High-Performance Computing Initiative, both Congress and the President have recognized the fundamental importance of the field. That recognition takes the form of a 30-percent increase in funding to $638 million this year. The administration has recommended to Congress an increase from last year of 100 percent over a five-year period, ultimately reaching more than $1 billion a year."

PARALLEL COMPUTING (p. 90). "Transaction processing on large databases is a bread-and-butter task for most large businesses. It has usually been accomplished on mainframes. Now, Ncube and Oracle say they have demonstrated the viability of massively parallel supercomputers for this mainstream business application. A 64-processor Ncube 2 running the Oracle Parallel Server database management system, version 6.2, has achieved a performance of 1,073 transactions per second on the Transaction Processing Performance Council Benchmark B.

"This performance was approximately twice that of any mainframe solution, said Michael Meirer, Ncube president. Cost per transaction was about one-twentieth of the cost of a high-end mainframe system. The achievement sounds like a foot in the door to another large application area."

Editor: Neville Holmes; neville.holmes@utas.edu.au

2008: The Year of Revitalization

Rangachar Kasturi
IEEE Computer Society 2008 President

Look for innovations in products and services that focus on career and professional development.

I extend a warm welcome and thanks to our members, volunteers, and staff worldwide. I thank you for your expression of confidence by electing me to lead the IEEE Computer Society as we move forward, building upon the progress made under the leadership of president Michael R. Williams and his executive committee during 2007.

Mike Williams led the Society through a year dominated by financial challenges and implemented many changes as part of an overall transformation. He initiated streamlining of the Society's governance structure and directed several new initiatives including the development of the Certified Software Development Associate credential to be launched in 2008 under the leadership of our Professional Practices Committee.

In 2007, we made substantial progress in recovering from our financial challenges. Although we continue to face a deficit, the ground has been laid for revenue growth in 2008.

PLANS FOR 2008

Every challenge brings with it opportunities. We see an opportunity to enhance the value of the IEEE CS to its members, its customers, the profession, and the public. Our revitalization depends on our ability to innovate, especially in those areas that enhance career development, and to discontinue products or services that do not contribute to our growth. We will fund several new initiatives without increasing the financial burden on our members or customers. Your suggestions are, of course, always welcome.

My plan is to shape 2008 as "the year of revitalization," a year when each of our program boards innovates new products that will pave the way for our prosperity. We have made great progress toward managing our expenses, but our recovery truly depends on growing revenues. Our director of finance and accounting has provided clear targets for operating margins over the next two to three years, and I have asked each program board vice president to develop plans toward achieving those targets.

In 2008 look for the completion of the new computer.org, built on a state-of-the art open source platform that will enable genuine member engagement; the launch of an entry-level software developer certification; an expansion of our highly successful conference publishing program to all of IEEE; a new Software Engineering Seminar Series; and a new version of the Computer Society Digital Library—the IEEE Computing Library—which will be targeted at software companies. All of these initiatives are being driven by what our members and customers have asked for: information and services that enhance career advancement, are easy to access, and that maintain the IEEE CS's reputation for the highest quality.

To ensure that we can deliver what you need, we are making some changes to the staff support structure. Seven functional groups are being created: Finance and Accounting, Information Technology and Services, Sales and Marketing, Business and Product Development, Products and Services, Governance, and Membership. We expect that this new organization will greatly enhance our ability to create new products and services.

I also plan to further streamline our governance structure. We're reducing the number of face-to-face meetings, but we are making the meetings more accessible to everyone. Boards and committees will use web and video conference tools to enable them to participate from their home locations, and video reports of the meetings will be posted on our website. I will also work with volunteer leaders to examine our existing board and committee structure to determine what activities could be effectively combined or reorganized.

2008 EXECUTIVE COMMITTEE AND BOARD OF GOVERNORS MEMBERS

It is with pleasure that I introduce the 2008 Executive Committee. The committee includes president-elect Susan (Kathy) Land, CSDP, principal systems and software engineer at the MITRE Corporation, and past president Michael R. Williams, professor emeritus of computer science at the University of Calgary.

George Cybenko, Dorothy and Walter Gramm Professor of Engineering and adjunct professor of

computer science, Thayer School of Engineering at Dartmouth College, is the first vice-president, Electronic Products and Services. Michel Israel, scientific counselor of the French Embassy, Washington, D.C., is the second vice-president and secretary. Joseph R. Bumblis, IT project manager at BAE Systems, is the vice-president, Technical and Conference Activities. Sorel Reisman, managing director of the MERLOT consortium and professor of information systems and decision science, California State University at Fullerton, is the vice-president, Publications.

Antonio Dória, project manager and business consultant at Enabler Wipro and chief software architect at Matakiterani; Stephen Seidman, dean, College of Natural Sciences and Mathematics, University of Central Arkansas; and John Walz, senior consultant at the Sutton Group, continue as vice-presidents of Chapters Activities, Educational Activities, and Standards Activities, respectively. Donald F. Shafer, chief technology officer at the Athens Group, will be Treasurer.

The three members of the IEEE Board of Directors elected by Society members—Thomas W. Williams, chief scientist at Synopsys; Deborah M. Cooper, founder and president of an independent consulting firm; and Stephen Diamond, president and CEO of Picosoft—serve as nonvoting members of the Executive Committee along with *Computer* editor in chief Carl K. Chang, professor and chair of the Department of Computer Science, Iowa State University, and the executive director, Angela Burgess.

I welcome Itaru Minura and Christina M. Schober, who return for their second terms as members of the Board of Governors, and our newly elected Board members, André Ivanov, Phillip A. Laplante, Jon G. Rokne, Ann E.K. Sobel, and Jeffrey M. Voas.

EXPRESSIONS OF APPRECIATION

I extend my gratitude to the following dedicated leaders for their many years of service to the Computer Society. The Society is indebted to them for their many valuable contributions. Those who completed their terms on the executive committee are Oscar N. Garcia, 2006-2007 IEEE Division V Director; Jon Rokne, vice-president, Publications; Stephanie M. White, vice-president, Technical Activities; and Christina M. Schober, secretary. Members who served on the Board of Governors whose terms expired in 2007 are Jean M. Bacon, George V. Cybenko, Antonio Dória, Richard A. Kemmerer, and Brian M. O'Connell. We look forward to working with them in future endeavors.

I also would like to take this opportunity to extend a special thank you to Oscar N. Garcia, who served ably as the IEEE Division V Director. Oscar was an effective ambassador for the Society in IEEE activities, and his support of the Society at the IEEE level was invaluable. His commitment to the Society set a standard by which all others will be measured.

I would like to close by thanking the IEEE Computer Society staff, particularly, for their dedication during a year of transition. They work tirelessly on behalf of members and volunteers, striving to further the Computer Society's mission. I especially acknowledge the contributions of the members of the Computer Society's executive staff, including Angela Burgess, executive director; Anne Marie Kelly, associate executive director; Violet S. Doan, director of administration; Neal Linson, acting director of information technology and services; John G. Miller, director of finance and accounting; and Richard J. Price, associate publisher.

Rangachar Kasturi *is the Douglas W. Hood Professor, Computer Science and Engineering, at the University of South Florida. Contact him at president@computer.org.*

Computer in 2008

Carl K. Chang
Iowa State University

There are numerous opportunities to volunteer for, participate in, and contribute to the development of *Computer*.

Carl K. Chang, Editor in Chief, Computer

As the flagship publication of the IEEE Computer Society, *Computer* publishes cutting-edge peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.

Because *Computer* must cover a broad spectrum of topical areas, articles featuring your favorite topics might not appear as often as you would like. If you feel that we have missed a major trend, you could help us by authoring a paper yourself, proposing a special issue, or simply writing to us. Together we are partners in building and sustaining a community that generates timely and pertinent high-quality technical content to help us maintain currency in the field and stay ahead in our careers. To appeal to *Computer*'s general readership, we must focus on practice or relevant research and present information in ways that are conducive to promoting understanding and applicability.

2007 COVERAGE

Topics covered in 2007 included network security, reconfigurable computing, data management, human-centered computing, multicore processors, 3D visualization, search innovations, Tablet PCs, embedded systems, service orientation, and green computing.

Let me explain again the difference between "special issues" and "theme issues." Special issues are typically guest edited by experts in the specific area who must submit a proposal that is subject to review and approval. Theme issues are a collection of articles on a related topic but not necessarily the result of a specific call. Special issues typically follow a well-defined development schedule targeted to a specific month, while a theme issue can be put together whenever it is ready and when the editorial space can be allocated.

In addition to special issues and theme issues, we also publish peer-reviewed Computing Practices, Perspectives, and Research Features. Therefore, I encourage potential authors to submit your manuscripts to *Computer* whenever you think they fit our editorial criteria and would appeal to this broad audience. Our acceptance rate is on the order of 20-25 percent cumulatively, and we do actively seek general submissions.

2008 OUTLOOK ISSUE AND BEYOND

As is our tradition, we open the New Year with our January Outlook issue, featuring forward-looking articles from across the computer science and engineering disciplines.

Michael A. Cusumano leads off with a fascinating analysis of the dramatic shift under way in the enterprise-software industry. In "The Changing Software Business: Moving from Products to Services," Cusumano observes that traditional product sales and license fees have declined, and product company revenues have shifted to services such as annual maintenance payments and technical support. A complicating factor is the rise of new business and pricing models such as software as a service and "free, but not free" software. It remains to be seen whether this is a life-cycle issue or a business choice, whether it is a temporary or permanent trend, but managers of software product companies nevertheless face a threefold challenge: managing the crossover, "servitizing" products, and "productizing" services.

In "Can Programming be Liberated, Period?" David Harel describes his now nine-year-old dream about freeing ourselves from the straightjackets of programming and of being able to move intuitively from "played-in" scenarios to running code. Quite a bit of work has been carried out since then, which, while still a far cry from justifying the replacement of a dream with a plan, now seems to offer some preliminary evidence of feasibility. The bottom line is this, says Harel: There is no reason why developers shouldn't make great efforts to bring widely researched and deeply worked-out ideas in computer science to bear upon the most basic and profound activity that involves computers, namely, programming them and running the resulting programs.

In "An Assessment of Integrated Digital Cellular Automata Architectures," Victor Zhirnov and colleagues examine cellular automata as a possible alternative to traditional von Neumann architectures as technology reaches the limits of CMOS and beyond. As computing

technology approaches the physical limits of scaling, the authors argue that it will naturally drive architectures of practical interest toward regular arrays of locally connected computational elements.

In "Toward a Competitive Pool-Playing Robot," Michael Greenspan and colleagues introduce Deep Green, a vision-based, intelligent robotic system developed to play competitive pool. It currently plays at a better-than-amateur level, planning and executing difficult combination and rail shots from across the table. It has pocketed runs of four consecutive balls, and it's only a matter of time before it can consistently run the table.

Nearly 150 years ago, Charles Darwin theorized how evolution and natural selection transformed the earliest life forms into the rich panoply of life seen today. In "Harnessing Digital Evolution," Philip McKinley and colleagues describe the dawn of evolution in a world we created: the world of computing. Digital evolution is a form of evolutionary computation in which self-replicating computer programs evolve within a user-defined computational environment. Over generations, natural selction can produce instruction sequences that can realize complex behaviors, sometimes revealing unexpected and strikingly clever strategies for solving problems.

Finally, in "Mining the Social Fabric of Archaic Urban Centers with Cultural Algorithms," Robert G. Reynolds and colleagues apply a suite of tools from artificial intelligence and data mining to existing archaeological data from Monte Albán, a prehistoric urban center. Specifically, the authors examine the period of occupation associated with the emergence of this early site, seeking to generate a set of decision rules using data-mining techniques and then using their cultural algorithm toolkit to express the underlying social interaction between the initial inhabitants.

Topics to be covered in other 2008 issues include Web 2.0, data-intensive computing, multicores, high-assurance service-oriented architectures, computational intelligence, mobile computing, cyberinfrastructure, e-science, and more.

APPRECIATION FOR SUSTAINED SERVICE

During my first year as editor in chief of *Computer*, I witnessed, with deep appreciation, the great expertise and commitment of editorial board members as well as the exceptionally professional and dedicated editorial staff members whose collaboration has resulted in outstanding issues, month after month.

As we have a policy limiting the term of service so that we can appoint new members to the editorial board on a rotating basis, the following editors have completed their dedicated service: Michael Blaha, area editor for databases/software; Jonathan Liu, area editor for networking; Michael Lutz, advisory panel member; Michael Macedonia, area editor for entertainment computing; and H. Dieter Rombach, area editor for software. Please join me in expressing appreciation to these dedicated volunteers for their service to *Computer* and to the Computer Society.

NEW EDITORIAL BOARD MEMBERS

To keep pace with rapid technology evolution, *Computer* must constantly build and rebuild its editorial board to ensure that it provides expertise in current and emerging technical areas. During 2007, I recruited Steven Reinhardt as area editor for computer architecture and Michael van Lent as area editor for entertainment computing. The following area editors will be joining the editorial board in 2008: Jean Bacon, distributed systems; Vladimir Getov, high-performance computing; and Sumi Helal, networking. Also, beginning with this issue, David Grier, formerly editor of the In Our Time column, will serve as editor of a new column, The Known World.

JOIN THE TEAM

There are numerous opportunities to volunteer for, participate in, and contribute to the development of each issue of *Computer*. Join the team in one or more of the following ways.

- *Submit your manuscript for consideration for publication.* Manuscript Central (https://mc.manuscriptcentral.com/cs-ieee), our totally electronic online service for processing manuscript submissions, provides complete author information and submission details.
- *Propose a special issue.* Contact Bill Schilit (schilit@computer.org), Special Issues Editor, to offer your suggestion or to receive information about submitting a special issue proposal.
- *Serve as a reviewer.* Indicate your interest in serving as a reviewer by sending an e-mail message containing your vita to computer-ma@computer.org.
- *Provide feedback.* We welcome your comments, and we encourage you to submit suggestions for topics to be covered in future issues of *Computer*. Send an e-mail message to chang@cs.iastate.edu.

We look forward to hearing from you, and we welcome your participation.

I wish you a wonderful 2008. Enjoy this Outlook issue and keep an eye on the upcoming issues throughout the year.

Carl K. Chang *is professor and chair of computer science at Iowa State University. He was the 2004 president of the IEEE Computer Society and formerly served as editor in chief of* IEEE Software *(1991-1994). Contact him at chang@iastate.edu.*

Internet Researchers Look to Wipe the Slate Clean

Sixto Ortiz Jr.

From online shopping to web-based communities to on-demand videos, the Internet has proven its mettle. Under the surface, though, the system, largely designed 45 years ago, is straining to cope with technology changes. These include faster network connections and microprocessors, and the availability of large amounts of inexpensive storage. The Internet is also struggling to support an increasing number of new uses it wasn't built for, such as mobile applications.

Researchers have come up with patches to let current Internet technology cope with new developments. However, experts say this is not an optimal solution and can't keep pace with technology advances.

With this in mind, scientists throughout the world are looking at a large-scale overhaul of some basic Internet elements to eliminate the need to constantly create workarounds to meet challenges. They are conducting what some call *clean-slate-Internet* research. The sidebars describe several major clean-slate projects.

The Internet's top three challenges are large-scale support for mobility, efficient content dissemination, and security, said Dipankar Raychaudhuri, a Rutgers University professor overseeing four clean-slate projects.

The new research could be expensive and might not yield widely

adopted results for 10 to 15 years. And overhauling the Internet could require the replacement of millions of dollars in existing networking equipment, as well as the development and implementation of new software.

However, proponents say the Internet has become so important in so many areas that it must be improved and modernized.

"[We should] analyze what we know about how the current Internet works, try to understand how the various applications it supports fare under the present architecture, identify areas of weakness, and produce one or more designs or design ideas to notably improve its functionality," said Internet pioneer Vinton G. Cerf, Google's vice president and chief Internet evangelist.

INTERNET SHORTCOMINGS

The initial research into the Internet's first iteration, the Arpanet, began in late 1962 at the US Defense Advanced Research Projects Agency. Central to this work was the development of packet-switching theory spearheaded by research groups at places such as the US's Massachusetts Institute of Technology and RAND Corp., and the UK's National Physical Laboratory.

The first Arpanet communication occurred in October 1969 between UCLA's Network Measurement Center and the Stanford Research Institute.

Taking advantage of new capabilities

Current Internet design doesn't work well with or take maximum advantage of several technological developments. In some cases, the technology available when the Internet was developed didn't allow researchers to give it capabilities that would be important today. Therefore, technology changes offer both an opportunity and a challenge.

Fast processors. Microprocessor performance has increased regularly for many years. Faster router processors could execute code within the network infrastructure on the fly and thus make the Internet more flexible, responsive, and "programmable."

However, manufacturers, still working with current Internet approaches, have not yet started taking advantage of the capabilities that faster router processors could enable.

Available storage. According to Rutgers' Raychaudhuri, today's IP technologies aren't designed to store data within the network. Because of this, vendors haven't designed Internet routers to store much data.

According to Raychaudhuri, the Internet needs protocols that incorporate storage.

New protocols, enabled by Internet routers that cache data, could avoid the problems that occur when TCP/IP times out sessions after a prolonged transmission disconnection. Overcoming this problem would minimize packet loss and the need for data retransmission, making content-related services more robust and efficient.

Published by the IEEE Computer Society

To address these issues, Rutgers University's Wireless Information Network Laboratory (Winlab) and the University of Massachusetts Amherst are developing *cache-and-forward* technology.

CNF delivers content efficiently to multiple mobile hosts by using a reliable hop-by-hop transport mechanism that stores an entire file in each node before forwarding it onward. Combining CNF with routers that have a large storage capacity would improve reliability because data stored in transit could be forwarded and delivered reliably after interrupted service is resumed.

According to Raychaudhuri, researchers have completed CNF's initial design and hope to have a prototype implementation running on testbeds by the middle of this year.

The Internet Research Task Force's Delay-Tolerant Networking Research Group is just starting to develop DTN technology. DTN would address the architectural and other principles needed to provide interoperable communications in performance-challenged environments such as spacecraft, battlefields, and disaster scenes, where end-to-end connectivity is not always available.

DTNs would use in-network storage to overcome transmission interruptions by resending stored data when connectivity resumes.

Programmability. Researchers are currently working on programmable network software that routers and other Internet hardware would run. Programmability could enable service customization in networks and take advantage of network virtualization, which is also being studied.

In network virtualization, the same hardware infrastructure would support multiple virtual networks. Each could have a different architecture or be optimized for a particular service, such as a specific type of security, or for a particular approach to services. This would make networks more useful, flexible, and efficient.

By executing network functionality and services in software, this approach would mark a departure in Internet technology.

Japan's New-Generation Network

Japan's Ministry of Internal Affairs and Communications formed the Study Group on Network Architecture in January 2007 "to discuss the need to create a network architecture based on new design concepts and technology and how R&D should be [pursued]," said Minoru Kubota of the Research and Development Office in the ministry's Information and Communications Policy Bureau.

The agency has initiated a project called the New-Generation Network. The Japanese National Institute of Information and Communication Technology is also working on the effort via its AKARI Architecture Design Project, noted Kubota.

The groups plan to establish a concept design by 2010, develop and verify the basic technology by 2015, and have the network running by 2020.

Carnegie Mellon's 100 × 100 Clean Slate Project

Carnegie Mellon University is leading a project named for the long-range goal of building a clean-slate Internet that will provide 100 megabit-per-second connections to 100 million US homes.

AT&T Labs-Research; the University of California, Berkeley; the Center for Appalachian Network Access; Fraser Research; the Internet2 Project; the Pittsburgh Supercomputing Center; Rice University; and Stanford University are also participating in the 100 x 100 Clean Slate Project (www.100x100network.org).

Economists, security and networking experts, network operators, and policy specialists are contributing to the project, designed to develop protocols and create an Internet that will offer large amounts of bandwidth and be economically self-sustaining, secure, and manageable. They will address issues such as network architecture, management, security, reliability, scalability, and economics.

The National Science Foundation has provided $7.5 million in funding for the project, which will entail fundamental research, proof-of-concept implementations, and testbeds that could be used in further studies.

Mission-critical uses

Today, the Internet is used for various mission-critical applications, such as e-commerce; monitoring power plants, factories, and equipment; and other tasks for which it was not designed.

One problem in particular is that the Internet currently lacks built-in strong authentication, which keeps unauthorized people from accessing important systems, said UCLA professor Leonard Kleinrock, an architect of the Internet's original packet-switching technology.

There are workarounds such as IPSec, which provides some authentication. However, Kleinrock said, patches add complexity and don't always overcome technical shortcomings, which frequently leads to ongoing patching.

Today's Internet, in particular TCP/IP, also does not work well with sensor networks, used in many critical monitoring and information-gathering systems. Sensors frequently experience latency, variable

US National Science Foundation

The National Science Foundation is running two major projects that support clean-slate-Internet research.

GENI

The NSF is planning an experimental research network known as the Global Environment for Network Innovations (www.geni.net). GENI will provide network facilities and equipment to help researchers worldwide experiment with new Internet designs.

Researchers could use GENI on projects such as new network-security models and applications that build their own communications stacks by programming portions of the network, explained Craig Partridge, BBN Technologies' chief scientist and the GENI Project Office's (GPO's) outreach director.

He said the network won't start operation for perhaps four or five years. The GPO is developing detailed engineering plans, and, if the NSF approves the funding, construction will begin. The GPO would then let academic and corporate research teams conduct work using slices of GENI.

GENI's early cost estimate is $350 million, Partridge said.

GENI will be a modular, expandable federation of programmable, highly instrumented data-communications platforms, including fiber-optic and wireless networks, and distributed clusters of systems, Partridge noted. Several working groups are determining exactly what types of equipment the network will use.

Researchers will access slices of particular platforms they want to work with, such as the fiber-optic or wireless systems, as well as the overall network's communications and computing capabilities, Partridge said.

FIND

The NSF is also funding 41 network-architecture-related projects through its Future Internet Network Design (FIND) initiative (www.nets-find.net).

These projects include the development of architectural support for network troubleshooting at the University of California, Berkeley's International Computer Science Institute; and research into monitoring, control, and troubleshooting mechanisms in future Internet architectures at the University of Pennsylvania, University of Massachusetts Amherst, and University of Minnesota.

response times, or intermittent connectivity, causing TCP/IP to drop important transmissions.

Security

Today's IP protocols were not designed for security, such as preventing spam or thwarting denial-of-service (DoS) attacks, according to Rutgers' Raychaudhuri, who is Winlab's director.

Over time, researchers added features to improve security such as digital certificates and encryption-based schemes like Secure Sockets Layer. However, these approaches require additional processing by network applications and hardware and thus add complexity and reduce performance.

IPv6, which the Internet Engineering Task defined in 1996 but which still is not widely used, includes some security features such as IPSec, which provides encryption and data validation. However, most experts say a more comprehensive solution is needed.

A big issue is authentication.

The Internet's early architects were trying to build a shared, open, and flexible network, based on mutual trust among community members, most of whom were fellow scientists, explained Kleinrock.

Thus, the Internet doesn't have the built-in ability to verify user identity or data integrity, via mechanisms such as authentication.

User authentication utilizes encryption, digital signatures, tokens, usernames and passwords, and other methods to verify that senders are who they say they are. This approach would help eliminate spoofing, which hackers frequently use in spam or DoS attacks.

Data authentication determines that no one has improperly altered or otherwise interfered with information. This is accomplished via methods such as checksums, message authentication codes, or digital signatures. These techniques help eliminate problems caused by malware and other code that hackers insert into messages.

According to Kleinrock, adding authentication would not be easy because patching legacy protocols is difficult, as is implementing patched or new protocols within the existing Internet infrastructure.

Mobility

Researchers designed TCP/IP for fixed computers with static IP addresses, said Raychaudhuri. Because early Internet-related protocols were built on a tight coupling between user, computer, location, and IP address, they don't work well with mobile users.

At the time, there was no meaningful mobile computing. Even as of October 2006, said the ClickZ Network, an interactive-marketing information provider, the portion of the online population that accessed the Web from a mobile device was only 19 percent in the US and 29 percent in Europe. Within the next five

to 10 years, though, 90 percent of Internet traffic will be among mobile devices rather than PCs, Raychaudhuri predicted.

To address wireless Internet access, the Internet Engineering Task Force developed the Dynamic Host Configuration Protocol, in which a server dynamically assigns IP addresses as needed to mobile and other devices that connect occasionally to the Internet; and the Mobile Internet Protocol, which lets wireless users roam from one network to another while maintaining a permanent IP address.

However, said Raychaudhuri, DHCP takes time to assign IP addresses, which causes connectivity delays. The approach also tears down sessions after each use, which means devices must spend time obtaining an address every time they try to connect to the Internet.

Mobile IP, he noted, is not widely implemented and has some problems.

For example, it can be inefficient. With Mobile IP, each enabled wireless machine has a home address—which remains the same regardless of the device's location—on a home network.

When away from home, the mobile unit sends its location information to its home agent. This is a router on the home network that tracks the device's location as identified by its temporary care-of address, assigned by the new hosting network's foreign agent.

The home agent associates each device's permanent home address with its care-of address so that users who want to communicate with the mobile node can still contact the same permanent home address, regardless of the device's location.

A data transmission is first sent to a device's home address and then tunneled via the home agent and foreign agent to the care-of address.

This process becomes inefficient and can lead to delays that cause problems for some applications if the sending device is far from its home network and thus must communicate

European Union's Future Internet Research and Experimentation

The European Union's Future Internet Research and Experimentation (FIRE) project is an experimental, Europe-based research initiative on clean-slate Internet concepts, protocols, and architectures that will address technological, industrial, and socioeconomic issues.

Research areas include the development of major, interconnected testbed platforms for experimenting with large projects, as well as the design of new architectures and protocols for increased scalability, complexity, and mobility.

Because of the scope of the current Internet infrastructure, researchers say development of a new network will be a long-term project.

Princeton University: PlanetLab

PlanetLab is a global research network for developing new network services and building the foundation of a future Internet.

For the past five years, about 1,000 researchers at academic institutions and industrial research labs have used PlanetLab (www.planet-lab.org) to work on new technologies in areas such as distributed storage, network mapping, and peer-to-peer systems. They have developed short-term experiments and long-term services.

The goal is to explore critical areas such as file sharing; in-network storage; content distribution; routing, multicast, and quality-of-service overlays; anomaly-detection mechanisms; and network measurement tools.

PlanetLab currently consists of about 800 nodes at about 400 sites worldwide (www.planet-lab.org/db/pub/sites.php). Officials hope the PlanetLab Consortium, managed by Princeton University, will grow to 1,000 widely distributed nodes that peer with the majority of the Internet's regional and long-haul backbones, as well as testbeds such as the US National Science Foundation's Global Environment for Network Innovations.

The network currently hosts about 600 active research projects.

Rutgers University's Winlab

Rutgers University researchers are working on four clean-slate Internet projects. The university's Wireless Information Network Laboratory (www.winlab.rutgers.edu) started the projects during the fall of 2006, and they are currently in the early to middle development stages, said professor and Winlab director Dipankar Raychaudhuri.

The projects include cache-and-forward technology; the CogNet protocol stack for adaptive networks of cognitive radios, in which a transceiver intelligently detects used and unused communication channels and sends data to the latter; and a geometric protocol stack—one in which layers provide well-defined services and abstractions to higher layers—for location-aware networking.

Stanford University's Clean Slate Program

Stanford University students, staff, and faculty from the Departments of Electrical Engineering, Computer Science, and Management Science and Engineering; and the Business and Law Schools are participating in the Clean Slate Program.

The stated goal of the project (http://cleanslate.stanford.edu) is to "reinvent the Internet" via long-term, unconventional initiatives to overcome security and mobility limitations, incorporate new technologies such as sensor and optical networks, and develop new types of applications and services.

Current projects include the study of wireless spectrum usage; fast, dynamic optical light paths; and enterprise network security. Stanford researchers are also developing the Rate Control Protocol, a congestion-mitigation algorithm designed to enable fast downloads.

Clean Slate researchers are working with private companies, particularly those nearby in Silicon Valley; collaborating with researchers at other institutions; and contributing to other major Internet-related initiatives such as the US National Science Foundation's Global Environment for Network Innovations.

over long distances to and from the home address.

Also, Raychaudhuri said, Mobile IP is designed to work with mobile clients only via a fixed gateway to the wired network. The technology is not designed to work with platforms that don't have a fixed gateway, he said. An example is a bus or plane housing a router that serves onboard wireless devices. Its router moves and changes its association with fixed networks, making it difficult for Mobile IP to work with.

Sensors

Many types of sensors are and will be connected to the Internet, said Guru Parulkar, executive director of Stanford University's Clean Slate Program.

These could include process sensors in a manufacturing environment that monitor variables such as temperature, pressure, and flow rates, as well as sensors that keep track of product inventories.

According to Google's Cerf, using these devices could be difficult with current technology because TCP/IP was not designed to deal with the frequent disruptions and latency in sensor systems. When TCP/IP encounters such problems, it ends the session.

Cerf said DTN might help with this by enabling communications even if there has been a transmission disruption or delay.

TRANSITION

Raychaudhuri predicted that multiple new Internet protocols, even those that handle the same functions in vastly different ways, will work at the same time and compete for users. He said the Internet could support these protocols via virtualization that lets multiple virtual networks run on the same infrastructure.

Running these virtual networks along with the current Internet would help users transition to new technologies.

Nonetheless, the transition would be tricky and potentially very expensive, particularly because so much legacy equipment is in use, said UCLA's Kleinrock. New technologies could require the replacement of routers, access points, servers, and client-level software.

Consequently, the transition to clean-slate Internet technologies will probably be slow, progressing from labs and prototypes to limited implementations for specific groups of users.

"The concept of clean-slate design is useful in the research phase," Kleinrock said, "but the best of those ideas need to be blended into today's fabric."

Cerf said clean-slate Internet technologies could be introduced in parallel with current approaches (as is occurring with IPv6 while most users are still working with IPv4), or a new network could be created using current technologies as scaffolding.

"We did this in 1983," explained Cerf, "using the NCP (Arpanet's Network Control Protocol) as scaffolding for the testing of TCP/IP."

Kleinrock said, "We need to alter certain aspects of the architecture but must do them in considered and thoughtful ways, with models, analysis, and measurements to back up the efficacy of the changes."

Cerf added that changes are likely to start in the near future because current Internet technology can't keep up. He said, "There is a good chance that a completely new networking design will emerge over the next two decades."

Sixto Ortiz Jr. is a freelance technology writer based in Spring, Texas. Contact him at sortiz1965@gmail.com.

Editor: Lee Garber, *Computer*, l.garber@computer.org

Providing Nanopower for Nanotechnology

A researcher has come up with a possible solution to one of nanotechnology's big challenges: finding tiny, easily accessible, affordable generators to power the ultrasmall devices that the approach could yield.

The output could one day power tiny devices "such as nanosensors or in-vivo biological sensors built on single nanowires or ultrathin semiconductor films. They could range in size from 100 nanometers to several microns," according to Georgia Tech University research scientist Xudong Wang.

The generators could also eventually power larger systems such as cardiac pacemakers, wireless sensors, or personal electronic devices such as digital-audio players or cellular phones.

Wang created a nanoscale generator that converts ultrasonic waves—high-frequency vibrations—into electricity. He fabricated the device with 50-nanometer-wide wires made of zinc oxide, an optical semiconducting material.

Other nanogenerator researchers had used this material, but their wires tangled in production. This was a problem because nanowires must have the same orientation for users to manipulate and integrate them into a device.

In addition, Wang explained, "For our nanogenerator, we need all the nanowire to generate electricity simultaneously and continuously, which can only be realized when all the nanowires are vertically standing and separated from each other."

Wang said his group is the first to grow precisely positioned, vertically aligned zinc oxide nanowires.

"To have all of them standing vertically, we needed to make sure their growth direction is perpendicular to the substrate," he explained. "This is controlled by many parameters such as the substrate, [wire-growth] catalyst, pressure, and temperature."

The Georgia Tech researchers used gold as the catalyst. Growing the wires at 900 to 1,000 degrees Celsius vaporized the zinc oxide and yielded droplets of a gold-zinc oxide alloy, from which the nanowires were grown.

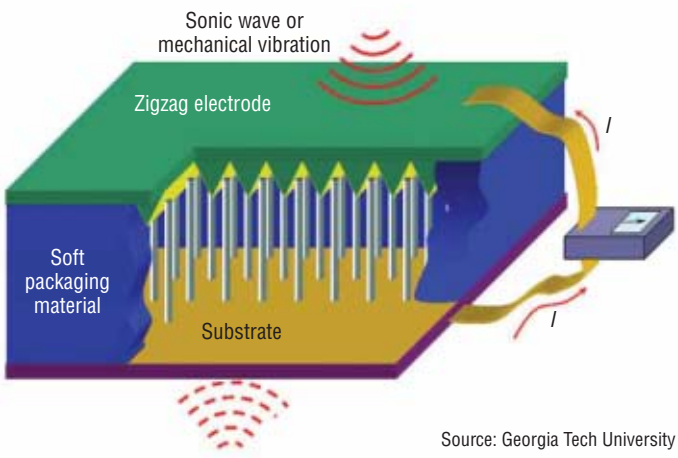
To generate electricity, Wang's team used forests of nanowire arrays on a gallium-nitride substrate—measuring between 30 and 50 nanometers wide and 1 to 2 microns long—that served as one electrode. They made the second electrode of platinum-coated silicon.

In one experiment, Wang and a colleague generated a very small current by bending the nanowires with the tip of an atomic-force microscope, which created an ultrasonic wave. In a different experiment, a wave generator produced the ultrasonic waves.

Either way, the waves squeezed the two electrodes together, causing the nanowires between them to flex. The wire's zinc oxide is a piezoelectric material. When flexed, Wang explained, strains form on the wire's sides, creating electric potential that generates a current when connected to an electronic circuit.

He said his team has increased the nanogenerator's output from an initial 0.6 to 0.7 nanoamperes to 600 nanoamperes, largely by increasing the number of nanowires and making them more uniform.

Wang said it could be three to five years before devices such as his generator are used commercially. ■



Source: Georgia Tech University

Georgia Tech University researchers have developed a tiny nanogenerator that could generate power for nanotechnology devices. The technology uses ultrasonic waves to squeeze two electrodes together, causing the nanowires between them to flex. When flexed, strains formed on the wire's side surfaces, creating electric potential that generates a current when connected to a circuit.

Company Releases Semantic Web Tool

A company has developed a beta version of a tool that uses advanced Semantic Web principles to help users better organize, explore, analyze, and share information.

Lew Tucker, Radar Networks' vice president of engineering, said Twine (www.twine.com) uses two Semantic Web technologies:

- the XML-based Resource Description Framework (RDF) for describing and interchanging metadata; and
- the Web Ontology Language (OWL), which improves the machine interpretation and processing of Web content by providing formal semantics that express relationships between objects within a domain.

These technologies are considered important parts of the Semantic Web, in which computers store data in machine-readable formats for easy retrieval, usage, sharing, and integration by applications, agents, and virtual assistants.

Radar Networks' tool either automatically places or lets users put information they gather online—from e-mail, web searches, PowerPoint presentations, videos, or documents—into a *twine*, which is basically an online topic-based knowledge-sharing network.

"People visit www.twine.com and can then add notes, pictures, documents to their own private collection," explained Tucker. They can then share information via twines. All the information is stored on Twine's servers except material the service points to on, for example, a video- or photo-sharing site.

For purposes of sharing information, users can designate whether the twines are open to the public or only to members of a specific group, noted Tucker.

Upon encountering an e-mail, text, or other file, the system creates a bookmark representation of the page, including metatags, title, and description. This makes it easier for other Twine participants to find the information when they use the application to conduct searches.

Twine automatically analyzes, classifies, tags, and identifies relationships between pieces of information using advanced machine-learning and natural-language processing algorithms.

In addition, Twine includes a graph that shows the relevancy and importance of information, Tucker said. Relevancy could be based on the way Twine users weigh their search, such as giving more credence to certain sources. As the researchers evolve the system, other properties—such as the number of links referring to a site, a potential indicator of its importance—may be used to prioritize search results, he noted.

The tool uses RDF and OWL to make the information it contains easier to understand, use, and share.

"This is a hot area in enterprise software," said Susan Feldman, research vice president for search and digital marketplace technologies with market-research firm IDC.

Vendors are developing many Semantic Web tools, but wide adoption will depend on increasing education and the understanding of semantic technologies more than on tool availability per se, noted Eric Axel Franzon, vice president of Semantic Universe, an organization that promotes awareness of semantic technologies. ■

News Briefs written by ***Linda Dailey Paulson****, a freelance technology writer based in Ventura, California. Contact her at ldpaulson@yahoo.com.*

Eliminating the Need for Antivirus Products

A researcher and software architect with the One Laptop per Child project has taken an innovative approach to eliminating the need for antivirus products in the computers that OLPC plans to sell for $100 or less to developing countries for use by needy children.

OLPC charged its director of security architecture, Ivan Krstić, with creating a secure system that would not require user intervention, a lot of technical support, or ongoing updates. According to Krstić, this would allow the laptops "to be usable by our youngest target audience: children as young as five."

He then built the Bitfrost security platform, named after Norse mythology's secure Bifröst bridge between the realm of the gods and the realm of mortals.

The platform is designed to dramatically raise the bar for would-be hackers and render many kinds of attacks useless, he explained.

Bitfrost seamlessly places every program on the Linux-based computer in a separate virtual operating system, extending an approach known as *container-based isolation*. This approach isolates a pro-

gram and keeps it from accessing or even being aware of other applications, the hard drive, or the main OS. Thus, problems with an individual program, such as viruses or spyware, can't cause difficulties elsewhere. This also neutralizes botnet threats, Krstić said.

In addition, the system lets programs operate on data but doesn't let hackers who attack an application exploit the information it contains.

OLPC has also implemented a mechanism that prevents hackers from changing the system's BIOS code, via a malicious program or even the operating-system kernel.

Security expert and University of Pennsylvania professor David J. Farber noted that he has ordered the system to evaluate it for himself. He said, "My casual observation is that virtualization works to a large degree." However, Farber added, he wants to check whether it is possible that malicious data could damage the virtual machine itself, which could then harm the rest of the computer.

OLPC, founded by MIT professor Nicholas Negroponte in 2005, has entered the mass-production phase. Uruguay placed the first order of 100,000 laptops, and initial deployment began in December 2007. ■

Editor: Lee Garber, *Computer*, l.garber@computer.org

Presenting the Princeton Laptop Orchestra

Carnegie Hall will be seeing a very unusual orchestra in April, one that uses laptops instead of musical instruments.

The Princeton Laptop Orchestra (http://plork.cs.princeton.edu), an ensemble that performs music via computers and specially designed speakers, plans to play the famous New York City concert hall with the conventional American Composers Orchestra on 25 April.

Orchestra members, composers, and conductors include Princeton University undergraduate and graduate students and faculty members. The musicians play 15 "instruments" consisting of identical hardware but different software and control devices.

They make music via sound synthesis and the signal processing of live audio input, explained Princeton assistant professor of music Dan Trueman, one of the group's founders, conductors, and composers.

Princeton University's Laptop Orchestra performs music via computers and specially designed speakers. The group has played in places such as Chicago and Washington, D.C., and will appear in New York City's Carnegie Hall in April.

The musicians work with a Bowed Sensor-Speaker Array—a combination of multichannel, omnidirectional speakers with sensors for audio control that Trueman helped design—as their sound source. Participants use a bow to play a fingerboard attached to the speakers.

The orchestra members also work with identically equipped Apple MacBooks and software based on the Max/MSP graphical programming environment, the SuperCollider programming environment and language for real-time audio synthesis and algorithmic composition, and the ChucK audio programming language.

When musicians play their instruments, the input data travels to a laptop that digitizes it and converts it into sound. The system uses signal processing and algorithms to synthesize the sounds of various acoustic instruments, said Trueman. It can also produce music via sound samples. The composers write applications that interpret the sensor data and map it to various types of sound.

Some orchestra members have used accelerometers inside the laptop that detect motion, such as the tilting of the machine, which the system uses to control volume, pitch, and other aspects of the music.

The orchestra has primarily played original music, written in standard musical notation or a unique graphic tablature.

The group has played in places such as Chicago and Washington, D.C. Former member Ge Wang is forming a laptop orchestra at Stanford University, where he now teaches.

Trueman, a classically trained violinist, said he wanted to take electronic music beyond the traditional confines of the studio by adding a live-performance component.

The Changing Software Business: Moving from Products to Services

Michael A. Cusumano
Massachusetts Institute of Technology

A dramatic shift is under way in the enterprise-software industry as established vendors embrace services in the wake of declining product revenues. It remains to be seen whether life-cycle dynamics or business-model choices are behind the long-term trend.

The dramatic changes in the software business over the past few years have important implications for both users and producers of software products and services. Traditional product sales and license fees have declined, and product company revenues have shifted to services[1] such as annual maintenance payments that entitle users to patches, minor upgrades, and often technical support.

This shift has been especially pronounced among enterprise-software vendors. We can clearly see this in the case of Siebel, whose product sales fell dramatically before Oracle acquired the company in 2005. A decade ago, even Oracle experienced the *crisscross*—service and maintenance revenues crossing over to exceed product revenues. We couldn't tell if Oracle and Siebel's product sales were dropping or product prices were falling, as Figure 1 depicts, but the effect was the same: Services (including maintenance, which typically accounts for up to 60 percent of service revenues) became more important than product revenues.

There are some exceptions. Product sales continue to account for most of game-software revenues, although online-gaming service revenues are growing fast. Platform companies like Microsoft—which has a large ecosystem of PC manufacturers as well as enterprise and individual users driving sales of Windows and Office—continue to generate enormous revenues from products. But even Microsoft is encountering change. The company reported that services in the server and tools segment accounted for about 3 percent of its fiscal year 2007 revenues and online services (MSN) for 5 percent of its revenues. Just a few years ago, Microsoft derived all its revenues from product sales.

A LONG-TERM TREND

Services' growing importance for software product firms dates back to at least 1990. The advent of free and open source software (which drove down software prices), as well as Y2K and the Internet boom and bust, accelerated the trend. In general, since 2000 or so, we've seen many enterprises and individual customers rebel against paying a lot of money for standardized or commodity-type software products.

New pricing models

A complicating factor is the rise of new business and pricing models such as software as a service (SaaS) and "free, but not free" software. Companies like Google, Yahoo!, and even Microsoft (with Windows Live and Office Live) now deliver what used to be for-fee software products ranging from search and e-mail to basic desktop applications as a nominally free service. The user doesn't directly pay for the software (unless you count the time to watch advertisements), but advertisers pay the software service vendor.

SaaS vendors such as Salesforce.com still count SaaS as product revenues, and keep them separate from professional services. However, the SaaS pricing model actually eliminates maintenance payments—a major source of service revenues for software companies—and often includes some bundled technical support—a source of costs. So the SaaS model has confused the traditional

Published by the IEEE Computer Society
0018-9162/08/$25.00 © 2008 IEEE

separation of product and service revenues as well as costs, and this should result in a decline in service revenues because of the elimination of maintenance payments.[2]

Life cycle or business choice?

What's happening to software product companies, especially those selling to enterprise customers, might be either a consequence of their life cycles or a business model choice to emphasize services more than product sales. The life-cycle idea suggests that software product companies start out generating most of their revenues from product license fees, but over time shift to a mixture of products and services and eventually to mostly services.

Firms might want to continue focusing on products because they can generate up to 99 percent gross margins, given that the marginal cost is zero to copy a piece of software or any other digital product. By contrast, margins for labor-intensive IT services can be 30 percent or lower.

As competitors appear, software product companies have trouble getting new customers, or are forced to lower prices due to competition from similar firms or free software. Then these companies are more subject to what I call the "99 percent of zero is zero" rule: The great profit opportunity from software products becomes theoretical and not practical. And, whether they like it or not, their revenues gradually shift to services.

There's more going on here than either an inevitable life-cycle effect or, in some cases, explicit managerial decisions to emphasize services more than products. On the one hand, if we look at other industries, usually in the beginning of their histories, we see a lot of attention paid to product innovation and design. Once companies get the product designs right or a dominant design emerges, they shift their emphasis to the process side, such as mass production, in a product-process life cycle.[3]

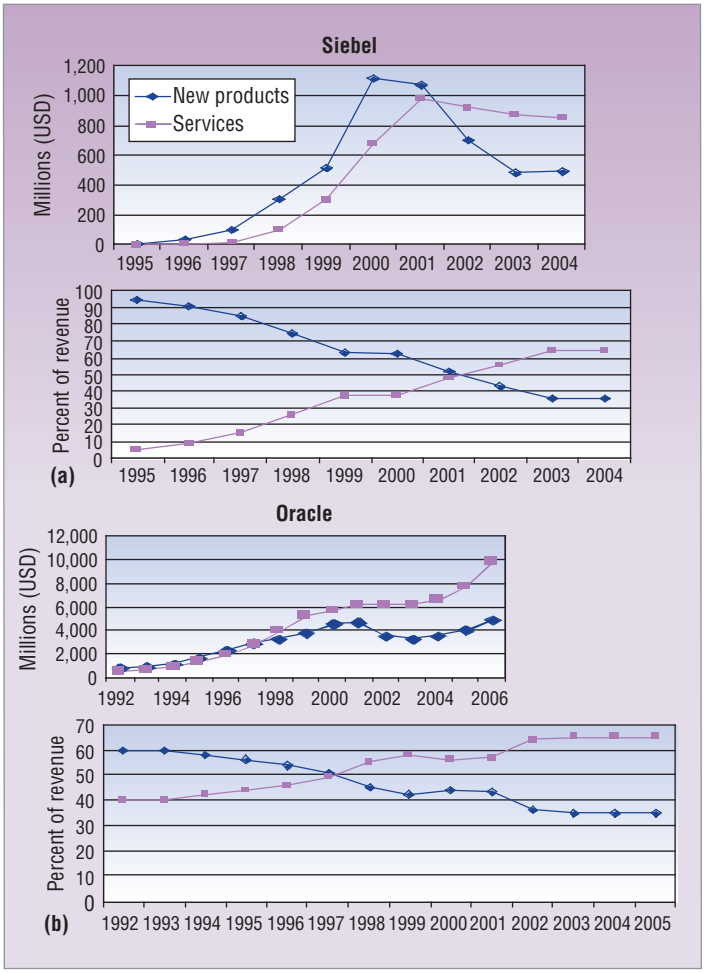


Figure 1. The crisscross. (a) Siebel's service revenue eclipsed products revenue in 2002. (b) Oracle's service revenue crisscrossed products revenue a decade ago.

Striving for efficiency

Firms aim for production efficiencies. In the early 1900s, Ford introduced the Model T (which became the standard automobile design), then focused on standardizing components and automating mass production. In the software industry, there's been a shift from product design in the 1960s to software engineering in the 1970s and 1980s, culminating in "software factories" in Japan and India, as well as the Capability Maturity Model in the US.

Service innovation is an aspect of the life cycle that might affect software and some other industries. For example, if the product design has become a commodity—widely available and low-priced around the world with little differentiation—and after a company has wrung maximum efficiency out of process improvement—then management might turn its attention to services.

On the other hand, what we're seeing might be related to "S-curves" and "disruptive technologies."[4] In software, not only do we have maturity setting in for different product segments and companies shifting their emphasis to services, but some new technologies now support different kinds of business models, including different ways of pricing and delivering software, and reaching different kinds of customers.

Obviously the Internet and wireless technologies enable all sorts of on-demand or transaction-based pricing models or Google types of advertising-based revenue models. In addition, a platform transition seems to generate demand not only for buying new products but also for services. For example, a customer switching platforms from mainframe to client-server or from client-server to the Internet or from stationary to mobile probably needs a lot of services in terms of strategic assistance,

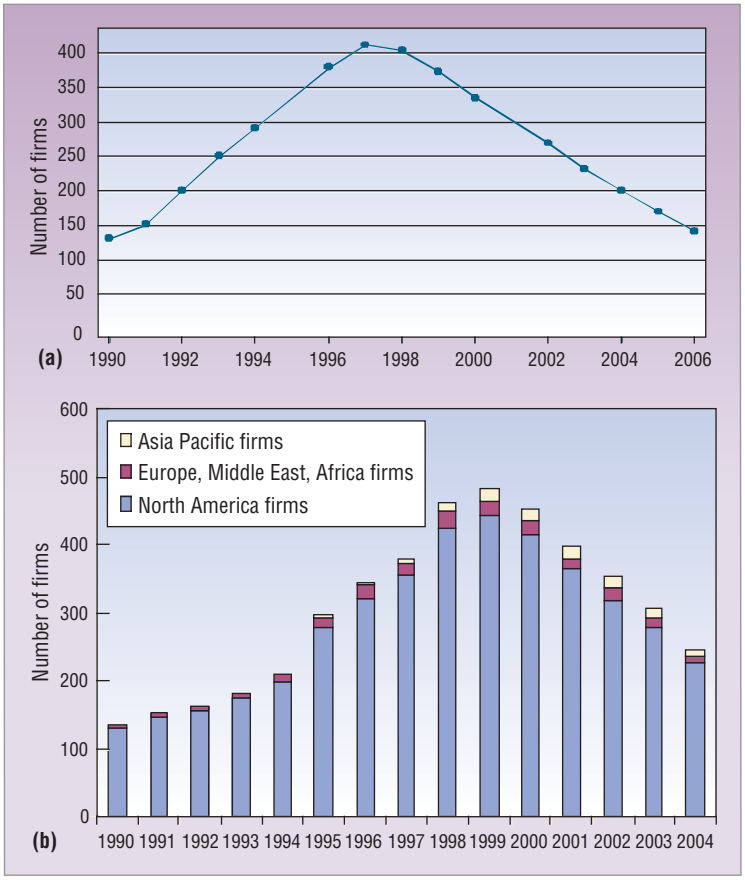


Figure 2. Industry growth. (a) The number of software product firms peaked in 1997 at about 400 before the industry underwent a rapid consolidation. (b) The number of IT services firms rose in the 1990s, peaking in 1999 at just below 500.

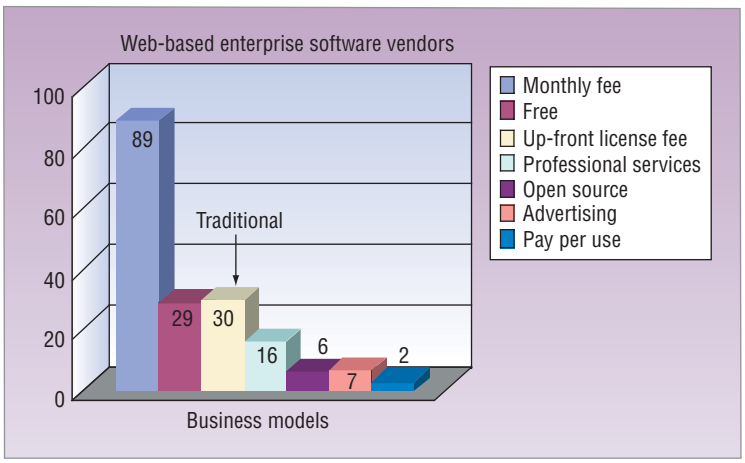


Figure 3. New business models. Web-based enterprise-software companies have adopted a variety of business models. Monthly subscription fees are the most popular pricing model.

rewriting applications and data, or retraining employees. In other words, platform transitions such as we've experienced over the past 15 years could also generate as much or more new revenue from services as from products, especially since many products are now free or low-priced.

SIMULTANEOUS MATURITY AND INNOVATION

To sort out what's happening in the software business, I launched a research project at MIT in 2003 to examine this shift from products to services for companies in software and other industries. My colleagues Fernando Suarez and Steven Kahl and I are still analyzing the data, but we have some preliminary findings and observations.

Peak and consolidation

The first database we created, covering 1990 through 2006, is a comprehensive list of firms that consider themselves software product companies selling "prepackaged software," listed under US Standard Industrial Classification (SIC) code 7372, as Figure 2a illustrates. This data includes foreign firms such as SAP and Business Objects that list on US stock exchanges, as well as game-software firms that sell products almost exclusively.

The data set contains about 500 distinct firms and peaked in 1997 at about 400 firms. By 2006, the list was down to fewer than 150 firms—indicating a dramatic consolidation of the software products business.

The second database, which covers 1990 through 2004, consists of firms that compete in IT services under several different SIC codes. This data, illustrated in Figure 2b, also shows listed companies peaking in 1999 at just below 500, and declining to less than 250 in 2004. The strong rise in IT services companies in the 1990s suggests that the transition from client-server to Internet platforms provided as many or more opportunities for services firms as it did for software product firms to become public companies, though both the services and products side of the business have experienced significant consolidation since that time.

The fact that the number of public software and IT services companies is consolidating suggests that the software business is maturing. However, other data collected at MIT suggests a strong rise in start-up enterprise-software companies, especially in 2005, using a variety of new business or pricing models (www.agoeldi.com/media/Thesis_AGoeldi_Final_09MAY07.pdf).

New business models

Figure 3 shows the business models of 108 companies

competing in Web-based enterprise software (about 20 percent of the companies are publicly listed), and indicates that monthly subscription fees are the most popular pricing model. A minority of companies also offered free software or advertising-based software (Google falls into this category), and others charged the traditional license fee.

Figure 4 shows a model my MIT students made that categorizes the variations now occurring in revenue or business models, delivery models, and target customers. A decade ago, nearly all software product companies sold software through the up-front license fee and did local installations on the customers' hardware. Now we have many different business models—subscription, advertising-based, transaction-based, and several kinds of "free, but not free."

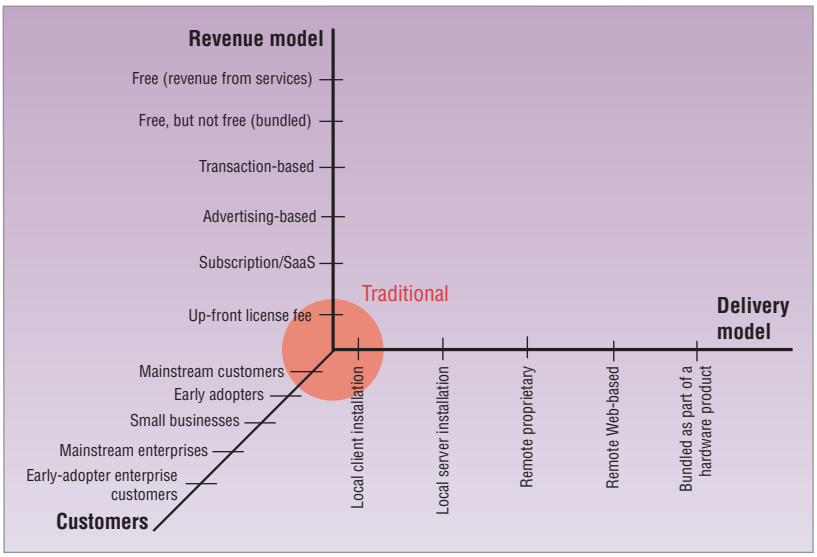


Figure 4. Business model dimensions. Companies have expanded their approaches in terms of customers and delivery and revenue models.

Software delivery models can be remote and web-based or bundled as hardware products. This trend toward potentially cheaper software, combined with less costly ways of delivering software over the Web, has made it possible for firms to target not only mainstream customers but small businesses and leading-edge early adopters.

In addition, many software companies are now turning into hardware companies in what's sometimes called the "appliance model" (http://dspace.mit.edu/handle/1721.1/39504). If you put the software in a box, it's less likely that the price will fall to zero. People usually will pay more for a box, even though they might not want to pay much for software or digital media on its own.

Another element behind this entrepreneurial activity is that it might take less money to start a software company. Of course, it was always possible for "two guys in a garage" to launch a software or computer-hardware company, and many started that way. But today, many critical enterprise components—the operating system, database, and web applications server—are available as free and open source software. An entrepreneur can write some applications code and then hire another firm to host the software and, with relatively little expense, launch an enterprise-software company. Data from a recent survey suggests that entrepreneurs funded about 37 percent of the new web-based enterprise start-ups, and only 36 percent relied on venture capital (www.agoeldi.com/media/Thesis_AGoeldi_Final_09MAY07.pdf).

Temporary or permanent?

As we look back at these trends and new developments in the software products business, a question occurs: Is this increase in services and new business models temporary or permanent? Permanent in my mind refers not necessarily to "forever" but to a trend lasting decades rather than years.

One possibility is that we're now merely in between platform transitions and probably at a bit of a plateau in terms of product revenue growth. If some major innovation occurs, such as for a new computing platform, then individuals and enterprises will again start buying new products, both hardware and software, in large numbers.

By contrast, the permanent argument says that software might have experienced what computer hardware did in the past: Investments from Intel and other firms along the lines of Moore's law helped dramatically reduce the price of computing power and bring powerful computers down to the level of commodities.

In other words, the permanent argument suggests that much software now is also commoditized, just like hardware, and prices will fall to zero or near zero for any kind of standardized product. In this scenario, the future is really free software, inexpensive SaaS, or "free, but not free" software, with some kind of indirect pricing model, like advertising—a Google-type of model. And it's possible that other commoditized high-tech industries, especially those with significant value coming from software, are likely to follow.

WHAT THE DATA SAYS

Perhaps we've experienced changes that are long-term, rather than temporary. But what does the data say? Our database of 500 publicly listed software product companies contains an average of about 10 years of data for each firm (totaling over 3,200 annual observations). Excluding game-software firms and some other firms

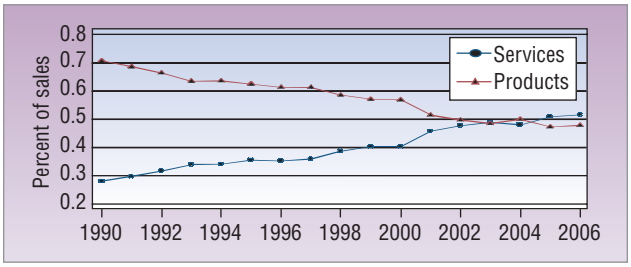


Figure 5. Industry crisscross. Software product firms overall saw product revenues decline from 70 percent in 1990 to less than 50 percent in 2003.

(mostly, they didn't break out products versus services and we couldn't confidently classify their revenues), the total number of firms peaked at 300 in 1997 and stood at merely 111 in 2006. As Figure 5 shows, software product firms in our sample had an average of 70 percent of their revenues coming from product sales in 1990 and less than 50 percent since around 2003, when the crisscross first happened for the industry as a whole. If we remove game-software companies from the sample, the crisscross happened in 2002 and is a bit more pronounced.

We didn't separate maintenance from other services because less than 10 percent of our sample broke this out. Firms treat maintenance as a type of service because, unlike with product sales, companies can recognize these revenues only as they deliver patches, upgrades, or technical support over time.

Some firms, such as SAP and Oracle, are now trying to relabel maintenance fees as product revenues in the sense that they represent product renewals. This makes some sense because maintenance has profit margins closer to product sales (though a bit lower because of the routine technical support costs usually included in the maintenance agreements), but maintenance revenues are still derived from the installed base of customers and recognized over time, like other services.

Reaching equilibrium

The data indicates that product revenues have dropped but haven't continued to fall to zero. Rather, they've stabilized at just over 50 percent of total revenues. So perhaps software product companies have reached a sort of equilibrium point as a business—more service (including maintenance) revenues from their existing customers than new-product revenues, but products are still holding significant value, at least for the publicly traded companies. Even without including game-software companies, we see this stabilization trend.

We can also look at how common it is for software product companies to sell only products as well as have different hybrid mixtures of products and services. In 1995, Richard Selby and I published a book that held Microsoft up as the ideal model for a software company—100 percent product revenues and those wonderful gross margins.[5] But the data suggests that these kinds of companies are relatively rare historically and are becoming fewer over time.

Our preliminary analysis also indicates that, while the average level of product revenues has dropped to less than 50 percent for the software product companies, the optimum mix for operating profitability (again, excluding games and some other firms) seems to be at about 70 percent products and 30 percent services. There are also some companies in our database that have reported 100 percent service (including maintenance) revenues in a given year and no product sales, even though they're nominally software product companies. Companies in this category are likely to be weak performers and candidates for takeover or bankruptcy.

Reasons for the shift

Why the shift toward services? On the surface, primarily it's happening because software product firms are getting older. They creep toward that service crisscross at the rate of nearly 2 percent a year. The crisscross point by age is at 26 years for the whole sample and 22 years if we exclude game companies. In other words, if a software product company survives for more than 20 years (and doesn't sell software games), it's likely that service and maintenance revenues now equal or exceed product revenues.

When we probe more deeply, statistical regression analyses suggest that this transition is also related to lagging growth in product sales and total sales, as well as the recession that followed the Internet boom. The appearance of the Internet as a disruptive new platform also generated new service sales, especially for IT services companies. But this factor is statistically less important than firm-level factors for the product firms, such as age and the lag in sales.

In other words, the shift toward services for product firms appears to have happened for two reasons. One is that product sales might continue to grow, but services grow faster, perhaps because price levels or the number of new customers falls. This situation is still relatively healthy, and firms can easily survive as hybrid businesses. The other scenario is that the products business collapses, and that's why firms cross over to a majority of service revenues.

This second scenario is potentially disastrous because it often means the firm must reorganize radically and perhaps quickly, as in the case of Siebel or another firm I've written about, i2 Technologies.[6] The firm can no longer support large product R&D groups with large marketing and sales expenditures. It must transition from designing products for a largely abstract set of users to building and servicing products for individual customers. Many firms don't make this transition or make it poorly and reluctantly, missing the opportunity to manage services as a strategic area.

IMPLICATIONS FOR PERFORMANCE AND STRATEGY

As we collected our data, my research colleagues and I thought the impact of rising services would have a negative impact on profitability and market value for a software product company because services tend to have lower profit margins and signal lower growth prospects. What we're seeing, however, is a more complex relationship.

For most software product companies, services generally contribute positively to their profits, but not in the linear manner we'd expected. More specifically, there seem to be "sweet spots" at the low and high ends of the spectrum. We can roughly say that, for the average software product company (excluding game software), services contribute positively to profits until they account for about 20 percent of total revenues.

After that point, services become a drag on profitability until they reach about 60 percent of revenues. Then services begin again to have a positive impact. One possible explanation for this curvilinear effect is that product companies might sign most of their customers to simple maintenance contracts for up to 20 percent or so of the retail price of their products, and these kinds of services are very profitable for them as long as technical support costs are minimal.

But as the product companies get deeper into labor-intensive services, such as product customization and complex integration work, or strategic consulting and training, services can become a drain on profits until the product companies gain enough scale and experience to perform these services efficiently. Then they begin again to make money from services, much like dedicated IT services companies do. SAP and Oracle would fit this model. Both are very profitable and have only about one-third of their revenues coming from new product license sales.

Market value, which generally tracks growth rates as well as profitability, follows a similar nonlinear curve. It seems to be positive until about 20 percent, then negative until about 80 percent, and then positive again. However, our data also indicates that, even in years when services positively contribute to profitability, market cap can drop as services rise. This suggests that investors still don't understand how important services have become to the revenues and profitability of software product companies.

Services as a strategic area

The positive impact of services on profitability and market value differs somewhat by product category, and we're still in the process of sorting out these differences. But the general conclusion seems to be that many or most software product firms can and should take advantage of services, especially maintenance, and not just let services "happen" because their product business declines.

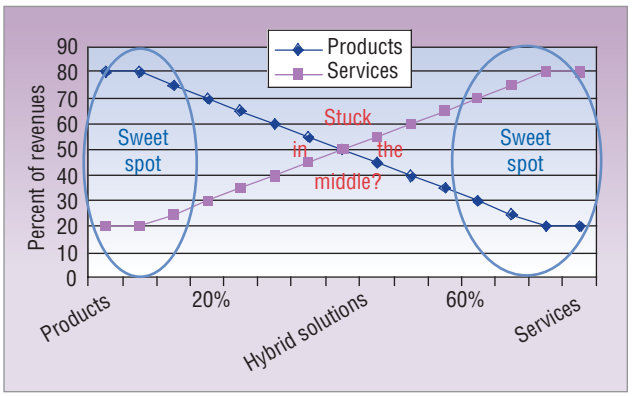


Figure 6. Sweet spots. Services contribute positively to profits at the low and high ends of the spectrum.

This means that software product firms—and probably many other high-technology firms—should treat services as a strategic area and a target of opportunity to increase revenues and profits—especially when the product business is suffering. We can see this in another preliminary analysis which suggests that, for every 10 percent increase in maintenance as a percentage of total services, service gross margins rise about 5 percent. In other words, if the products business is declining and shifting to services, companies should try to sign every customer to a maintenance agreement to minimize the impact on profitability.

By contrast, too many product firms seem to treat services as a necessary evil and manage them as a cost center, without much creativity or effort to grow that part of the business. In fact, though, most firms can look at their past trends and predict when they'll hit the crisscross and take some strategic responses, such as trying to rejuvenate the product lineup or launching a major campaign to sell more maintenance and other services, as firms such as SAP and Oracle have done over the past decade.

We also found that this trend toward services isn't limited to the software business, though it seems to be less of a life-cycle phenomenon and more a strategic move in other closely related industries, such as computer and telecommunications hardware and equipment. For example, as Figure 7 shows, IBM's service revenues have grown from less than 30 percent of revenues to more than 50 percent over the past decade. Sun Microsystems, Hewlett-Packard, Cisco, and even Dell have shown major increases in services and this seems to correspond to the commoditization trend in hardware.

Effect on IT services firms

The shift toward services for the product companies might be bad news for the dedicated IT services companies. Firms such as Accenture and Infosys are historically partners of enterprise-software product companies like SAP and Oracle, and they gain significant revenue by

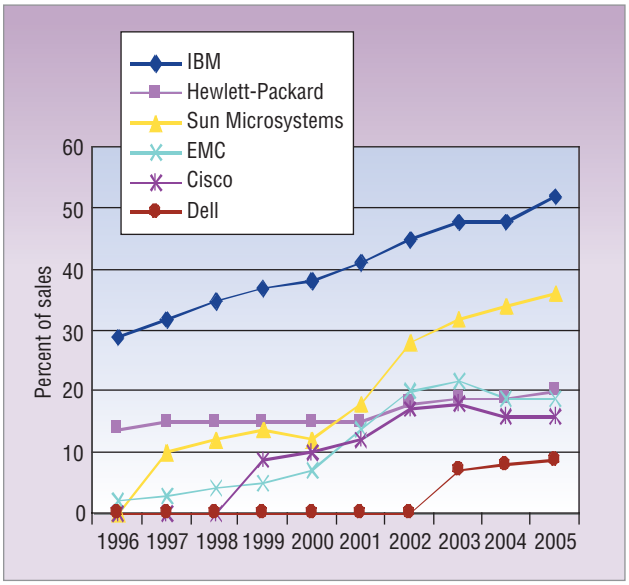


Figure 7. Hardware-company trends. Major hardware players have experienced varying increases in service revenues over the past decade.

installing, integrating, and customizing enterprise systems. But services are really money that product companies "left on the table" in the hope that services partners can help them sell more products. If the product revenues disappear, however, then former partners must compete for the same money.

THE THREEFOLD CHALLENGE

There's a threefold challenge for managers of software product companies and other firms experiencing this shift toward services.

Managing the crisscross

First, how can you manage this crisscross? Managers need to identify the best mix of product revenues (hardware and software, if appropriate) for their particular business segments along with service and maintenance revenues and determine how to impact these percentages. Services seem especially complementary in some business segments, like enterprise applications, while they're potentially more of a drag on other segments, although recurring maintenance payments are probably good for every product company.

Another point we tend to forget is that, for most product companies, products are the engine that drives service and maintenance revenues. Products and services are coupled for most firms, even though IBM and a few other companies such as General Electric have managed to become relatively neutral vendors of services. Most product firms need to maintain strong product lineups that keep customers paying for implementation or strategic services as well as long-term maintenance contracts or subscriptions.

'Servitizing' products

Second, managers need to think about how they can "servitize" their products—that is, create service offerings that add value and distinctiveness to their products. Services wrapped around products can make the products less commodity-like as well as generate new revenues and profits, even as the product business declines. In some industries, there's evidence that services over the lifetime of the product can generate several times the initial profits on the sale.[6]

Some day soon, for example, companies will give away various devices for free and just sell services or some kind of subscription contract. The cell-phone industry is well on the way toward this path. The automobile industry might follow as well. Even today, General Motors and Ford make little or no money from their products business while nearly all their revenue comes from financial services such as loans and leasing.

In the automobile industry, other ecosystem players make even more money from insurance and other services. What GM and other distressed automobile companies should do is give away their products at cost and sign customers to all-inclusive lifetime services contracts—not only loans or leases, but also insurance, maintenance and repair, and telematics services like GM's OnStar.

'Productizing' services

Third, managers need to think about how to "productize" their services so they can deliver them more efficiently. Productization of services can come from component or design reuse, computer-aided tools, and standardized process frameworks and training, as seen in past Japanese software factories such as at Hitachi or Toshiba, or in present Indian IT services companies such as Tata Consulting Services, Wipro, and Infosys. But productization can also come through automating services, such as the way eBay, eTrade, Expedia, Google, Lending Tree, and other Internet companies deliver their software-driven products or services.

In fact, fully automated services should be able to generate the same level of gross margins as a traditional software product company. That's why web-based delivery of software that different business models support is such an intriguing change for how producers distribute, deploy, and receive payment (or don't receive payment) for their software products and services. It's also why Google now rivals Microsoft in profitability, market value, and leadership in the software business.

In the future, as my colleagues, students, and I continue to do these kinds of analyses, we probably will change the way we think about the software business and some other high-tech sectors like Internet services, telecommunications, and digital media. There will probably always be some traditional product companies like

Microsoft that package technology and sell thousands or even millions of copies of their products. But our data suggests that not only are the numbers of these companies dwindling, the survivors also have to spend a fortune on sales and advertising as well as product development. As a result, most traditional software product companies make little or no money for their investors, and that's another reason why the smaller firms are disappearing. We would get a different picture, however, if we included companies like Google (whose SIC code lists it as an Internet services company) and perhaps some of the new SaaS start-up companies in the ranks of software product companies. Combining this data would give us a better idea of how much money customers are actually spending (directly or indirectly) on software-*based* products and services rather than just traditional software products—that is, including automated, standardized services and digital content delivered over the Web.

Acknowledgments

Thanks to the following MIT students for supplying figures: Francois de Laigue for Figure 2b, Andreas Goeldi for Figure 3, and the team of Krishna Boppana, Andreas Goeldi, Bettina Hein, Paul Hsu, and Tim Jones for Figure 4.

References

1. M.A. Cusumano, *The Business of Software*, Free Press/Simon & Schuster, 2004.
2. M.A. Cusumano, "The Changing Labyrinth of Software Pricing," *Comm. ACM*, July 2007, pp. 19-22.
3. J. Utterback, *Mastering the Dynamics of Innovation*, Harvard Business School Press, 1994.
4. C.M. Christensen, *The Innovator's Dilemma*, Harvard Business School Press, 1997.
5. M.A. Cusumano and R.W. Selby, *Microsoft Secrets*, Free Press/Simon & Schuster, 1995.
6. T. Knecht, R. Leszinski, and F.A. Weber, "Memo to a CEO: Making Profits after the Sale," *McKinsey Quarterly*, Nov. 1993, pp. 79-86.

Michael A. Cusumano *is the Sloan Management Review Distinguished Professor at the Massachusetts Institute of Technology's Sloan School of Management and Engineering Systems Division. He specializes in strategy, product development, and entrepreneurship in the computer software industry, as well as automobiles and consumer electronics. Cusumano received a PhD in history and East Asian languages from Harvard University and completed a postdoctoral fellowship in production and operations management at Harvard Business School. Contact him at cusumano@mit.edu.*

Can Programming Be Liberated, Period?

David Harel
Weizmann Institute of Science

The author describes his dream about freeing ourselves from the straightjackets of programming, making the process of getting computers to do what we want intuitive, natural, and also fun. He recommends harnessing the great power of computing and transforming a natural and almost playful means of programming so that it becomes fully operational and machine-doable.

Nine years ago, I sat down to write about a dream, one that would allow us to go from intuitively "played-in" scenarios to running code. Some of its most technically challenging parts were stated without providing too much support for their feasibility. Hence the choice of the term "dream." Ever since that paper was first published in 2000,[1] not only hasn't the dream evaporated, but it has continued to have a nagging presence, looming even larger in my mind, and getting broader and more elaborate by the year.

More significant is the fact that quite a bit of work has been carried out since then, which, while still a far cry from justifying the replacement of a dream by a plan, does now seem to offer some preliminary evidence of feasibility. Consequently, I've decided to revisit the topic and to describe the dream anew, or, more correctly (but possibly not very wisely), to propose a more dramatic and sweeping version thereof.

I should apologize to the reader at the start that this article doesn't get very specific or technical at all. Moreover, with the exception of the sidebar, it might read like the ramblings of a crazed, or dazed, individual. I should also point out that this article's title is, of course, intended to be a catchy take on the title of John Backus's wonderful Turing Award lecture and paper, "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs."[2]

PROGRAMMING'S STRAIGHTJACKETS

We've come a long way since programming had to be done by tediously listing machine-level instructions that prescribed how a specific computer was to modify and move bits and words in its memory. It's not my intention to attempt a survey of the history of programming. Still, it's obvious that there has been an amazing transition up the language-generation ladder, from machine languages to assembly languages, then to conventional imperative programming languages, and from there to the variety of contemporary programming styles—functional, logical, concurrent, visual, synchronous, constraint, object-oriented, aspect-oriented, and on and on. And there also have been numerous special-purpose languages, constructed for specific kinds of applications.

However, there is a sense in which programming is still the same kind of technically tedious task, albeit carried out on a higher, more appropriate, level of abstraction. It still entails writing programs, usually by using symbols, keywords, and operational instructions to tell the computer what we want it to do. A compiler is but a means to translate in reverse, down the generation ladder, rendering high-level programs readable and executable by the machine. And programming still requires testing and debugging, or preferably verification, to make sure that what we told our computer to do will have the results we desire. Moreover, we also must carefully specify the very

Published by the IEEE Computer Society
0018-9162/08/$25.00 © 2008 IEEE

concept of what we desire, which often is as complex and as error-prone as the program itself.

In the present time and age, there is an additional complication, which stems from the ever-increasing number of applications that involve multiple "pieces" that operate concurrently and are often widely distributed. Just think of Internet applications and web services, for example. For these, the added issue involves having to program each part separately, and then having to make sure that the compound behavior, the orchestration of the parts, results in what we want.

I submit that, by and large, even the most modern approaches to programming still suffer from these constraints, which we might term the "three straightjackets of programming":

- (I) the need to write down a program as a symbolic, textual, or graphical artifact;
- (II) the need to specify requirements (the what) separately from the program (the how) and to pit one against the other; and
- (III) the need to structure behavior according to the system's structure, providing each piece or object with its full behavior.

Can we liberate programming from the keyboard, from having to specify both the what and the how, and from the system's structure?

Can we liberate programming from these three constraints: from the keyboard, from the thankless tension between the what and the how, and from having to partition the dynamics along the lines of the structure?

But what is the alternative, we might ask. How can we program a computer without telling it exactly what to do, and without having to use a tangible medium to inscribe that telling? How can we ever be sure that what we get is what we wanted unless we state both and then compare them? And how can we program a multitude of things, other than by giving each thing its own instructions?

Of course, there are entire approaches to programming for which the explicit intent is to remove or alleviate some of these constraints. Thus, for (I) above, researchers have developed novel techniques in certain specific application areas to decrease the effort involved in writing the relevant programs; query-by-example for relational databases is such a technique,[3] as are spreadsheets.

For (II), logical and functional languages,[2,4] as well as many attempts at what has been generically called automatic programming, such as the particularly ambitious and interesting idea of intentional programming,[5] are all intended to allow us to state what we want, with much of the how-it's-done left to the compiler or interpreter. The intent is similar for constraint-based languages[6] and special-purpose application generators. And for (III), the recent wave of aspect-orientation[7] tries to ease the need to totally align behavior with structure by making it possible to supplement the internal behaviors of objects with special kinds of cross-cutting behaviors that weave through several of them.

My dream is a lot more ambitious. I am much greedier, and want it all. Can we not, I ask, push the envelope on all of these, and in the process try to change the face of programming? Not just by a new generation of higher-level languages or an innovative methodology, but by approaching programming quite differently. My dream is about freeing ourselves from the straightjackets of programming, doing our work in a far more liberated way, making the process of getting computers to do what we want intuitive, natural, and also fun. I move that we harness the great power of computing to help in this very quest, in making a far more profound downward transition than that embodied in compilation, taking a natural and almost playful means of programming and transforming it so that it becomes fully operational and machine-doable.

And lest you are thinking, "Okay, here comes another one of those people—a guy with a magical solution to all problems," I should add: No, I don't have a solution. I don't have anything that works for all kinds of programming, and I definitely have nothing that is magical. However, some preliminary evidence indicates that the yellow brick road might be worth a walk, at least for a certain type of program.[8] But the message here is definitely "maybe we should be thinking about this some more," not "if you just do it my way you'll be fine."

THE DREAM, PART 1: PLAYING IN THE PROGRAM

Programming is not about doing; it's about *causing* the doing. We "program" all the time, although not necessarily computers. We get (or try to get) other people to do what we want, and we guide them to behave in ways we approve of. We bring up our children, we supervise underlings, and we run companies, departments, and faculties. We make sure (or try to make sure) that our stockbroker, our handyperson, and our real estate agent do what we want.

We achieve these things by issuing explicit instructions when needed. However, more importantly, this usually requires a combination of laying out general principles, showing or walking through examples of what we have in mind (or *being* an example), and prescribing rules and conditions for what can or must be done versus what must not or cannot be done. Increasingly, we rely on the accumulated abilities of the person being "programmed" to abide by this guidance (and of course on that person's willingness and integrity—or fear of the repercussions). To varying degrees, organizations and governments, as well as religions, also work that way,

getting people to be good citizens. Notice that restrictions and constraints are a natural and crucial part of this "programming." If it is important that something be done, no matter what, or that something is never done, we simply say so explicitly (or the book of regulations says so, or the Bible says so, or whatever), and the person doing the doing must comply.

My dream is to be able to program computers that way too. I'd like for us to be able to remove the double quotes from the previous paragraphs, to substitute computers and computational devices for people and organizations, and to find similar ways to make machines do what we want. Ways that come naturally to us and are a smooth extension of the way we think; ways that require far less technical prowess than today's programmers need, and which allow flexibility on the computer's part in achieving the goals we have set out while honoring our requirements and making sure not to violate any of our constraints.

So far, this sounds like an illusion—worse, a hallucination—rather than a good old solid dream. Let me try to put some flesh on it by talking about actual computers.

Although they are not quite like humans (notice the omission of the word "yet"), computers are coming along in leaps and bounds. It is hard to guess what the world of computing will look like in, say, 20 years, but we are already seeing amazing progress in language and voice recognition, vision, human-machine interfaces, logical and deductive abilities, heuristic reasoning, and much more. And all this without even mentioning the Web and the way it's changing so many of our conceptions about computers and computing.

In many cases, the mathematics and algorithmics that underlie things computers do are getting more deeply buried inside, far from the user and often even from the programmer. And that's the way it should be, I claim, just like the way calculations underlying a spreadsheet are hidden from its user. In fact, the very borderline between user and programmer is becoming blurred.

Increasingly, computing calls for having to program incredibly complex *reactive systems*,[9] rather than systems whose role is to carry out numerous calculations. These are highly dynamic, discrete, event-driven systems, often with stringent timing constraints.

A reactive system's complexity is far less a result of complex computation and heavy algorithmics or the need to explore and mine intricate data. Rather, the system's complexity is a result of its subtle and complex (and often unpredictable) interactions with its environment and among the various parts of the system. These interactions consist of triggered and triggering events, changes in values, time-related constraints, probabilistic decisions, and so on, potentially happening in parallel in synchronous or asynchronous ways.

I believe that, as a general family of challenges for the world of computing, reactive systems are not only the hardest and most complex but also those in which centrality and significance will only increase. Again, no divine prophecy is required to see this; we only need to take an educated look at this Web-dominated, computers-are-everywhere era. Hence, although there are many significant kinds of nonreactive systems, and similar dreams might be articulated for them too, I dream about reactivity, to which the bulk of this article is devoted.

I claim that current methods for dealing with programming the dynamics of reactivity, however powerful and convenient, suffer from the same woes: We sit in front of a screen and write (or draw) programs that prescribe the behavior for each of the relevant parts of the system over time. Then we must check/test/verify that the combined behavior of all the parts satisfies a separately specified set of requirements or constraints. I dream of being able to do this quite differently.

Reactive systems are not only the hardest and most complex but also those in which centrality and significance will only increase.

Suppose you want to program a mobile phone. In fact, to make the story a little more direct, let's assume you're holding the phone in your hand, but that it's not a phone yet. It looks like one—it has a display screen, standard phone keys, four or five additional buttons, a port for communicating with the cellular antennas, and so on.

Let's further assume that you know the basic separate capabilities of each of these features. For example, the user can press and release a key, the device's internal illumination can be on, and so forth. The user can't manipulate the display, but the display can be on or off, show alphanumeric characters, display graphics and animations, and so on. However, other than knowing about these objects and their local capabilities, this is not yet a phone at all. It hasn't yet been endowed with phone-like behavior; you press a key, for example, and nothing happens.

What would be the most natural way to "teach" the device to be a phone? If you could talk to it, like you talk to a child or a student you are supervising, how would you proceed?

Well, as a start, you might say to it something like this: "Hey phone, whenever I press this key and hold it down for at least a half-second, you should switch on—meaning that your display should light up and show the cellular provider, my name, and the time." You might then give it similar instructions for switching off, possibly adding something like, "And, by the way, despite what you've just heard, don't ever switch off if the display shows that a text message is still in the process of being sent."

I should remark here that for the sake of this article, the example has been made rather simple. In actuality, we might also want to include in the switching on and off the starting and stopping of the communication between the phone's port and the cellular antenna. The protocols for these communication exchanges would have to be specified too.

You might next decide to start dealing with calls, saying something like, "Here's an example of how I'd like to make a call. If I press between three and 12 numeric keys sequentially, and then I press the green send key, you, in response, are to send out a call request to the cellular antenna containing the number formed by the pressed keys." To this you might add, "But if any two keys are depressed at the same time, you do nothing."

I'm not saying that this is *exactly* how I dream of programming a cell phone, but it's not that far off. Here are some relevant points.

Play-in is intended to constitute a natural and smooth computerization of how we'd cause some entity to behave the way we want.

First of all, if we don't have a phone, there's no point in trying to talk to it. Rather, we would be dealing with a computerized *mock-up image* of the phone, say, as a GUI on a computer screen. If we already have a graphical design for the phone, this would allow us to lay out the various objects as they would appear on the real thing; if not, we could show them in abstract form, say, as a structure diagram or object diagram. It would also make it easy to include in the process internal objects that the phone's user won't normally see, as well as nontangible objects that no one will normally see. Moreover, if we did have a real physical phone, but devoid of behavior, I can imagine working opposite it with no need for a soft mock-up of any kind.

Second, this process is about communicating to the system, in a natural style, the various pieces of behavior that we are interested in (or examples thereof) and teaching the system how to participate in them. These slices or chunks of behavior are not homogeneous in relation to the programmed system's desired overall behavior. They are *multimodal*; they can be specified to be a mandatory part of the behavior or a conditional part; they can be forbidden or preferred; they can be probabilistic, nondeterministic, or time-controlled; and so on. My point is that all of these are legitimate parts of the programming process—just like our telling someone what they can or cannot do, or when and under what conditions is part of our "programming" them.

Third, although advances in speech recognition and natural-language processing might eventually make it possible to freely talk to a phone or to its onscreen image, the sample session above is not about talking; it is about walking. That is, the process of realistically walking the system through the scenarios, hand-holding it while we show it, in a manner of speaking, how to cross a busy road without getting killed.

The term I have used for this in the past is *play-in*.[5] The point is not to talk about manipulating keys and displays, but to actually do the manipulation ourselves, in much the same way we expect to do it when the phone is built and we are *using* it rather than programming it. Rather than saying to the phone, "When I press this key …" the programmer would actually do it, for example, by clicking its onscreen image. Instead of saying, "You now make this light come on," we would actually show the phone what it should do by turning the light on—say, by selecting this action from a list of the light's capabilities.

Anything done in this way is done on the screen, or with the physical behavior-free system, in exactly the way we would like to see it done in the final programmed system. Representative examples, such as placing a call, would also be played in directly. The programmer would do this in a generic, by-example mode—just like the parent or educator—playing in an actual example of dialing a number, taking care to differentiate the example parts of the behavior from the fixed ones, and making the appropriate links—for example, the sample number dialed would have to be linked to the one sent out to the antenna.

Fourth, while many systems lend themselves nicely to GUI-based rendition, there is no a priori reason why we cannot carry out a similarly intelligent tutoring-like play-in process for systems whose front end is less discrete and less rigid. I am sure that experts on human-computer interaction would be able to come up with all sorts of analogs of the click, drag, and menu-select actions we do on GUI objects, which would work for playing in the behavior of more dynamically animated systems, such as games, navigation systems, automotive systems, tactical and avionics simulations, and so on. Thus, hybrid systems, which mix the discrete with the continuous and stochastic, are a special challenge here. Note, of course, that the better human-machine interfaces get, the richer play-in can become. A good example would be the ideas in Microsoft's experimental tabletop computing system.

Notice that I've said nothing yet about how to *run* "programs," only about how to "write" them. Nevertheless, it might make sense to pause here for a moment and see how this kind of intelligent play-in avoids the three main woes of programming—at least for the dynamics of reactive systems.

For issue (I), play-in is a walkthrough-style guiding, teaching, coaching, constraining process, playfully interactive, that is carried out directly and visually with the system's external or internal interface, and it does not

require the programmer to sit down and prepare a complete artifact of any formal kind.

For issue (II), both the what and the how are equally valuable parts of the play-in process, which can contain as much of either as the programmer decides. There is no need for separate specifications for the operational tasks and the requirements thereof. Anything that falls inside the total sum of what has been played-in will be a legal behavior of the system.

And finally, for issue (III) there is absolutely no requirement to capture or specify behavior per object/part/piece/chunk. On the contrary, I believe that the dynamic and interactive style of play-in encourages specifying interobject, or interpiece, behaviors whenever possible. But in any case, we should be free to specify behaviors or behavioral rules or constraints any way we want, even if they are, in our minds, orthogonal to the system's structure.

Play-out means employing powerful computing transparently to execute the grand total of all played-in pieces of behavior.

Incidentally, removing limitation (II), about separate whats and hows, also helps to deal with the classical question of completeness, that is, figuring out when we've finished the programming. The reason is that one way of describing play-in is that we can use it to program in the requirements directly, as part of the program. When we've finished doing that, we can be sure that nothing important has been left out, unless the requirements document left things out, something that in general no one can discover.

Also, regarding limitation (III), once we have liberated the programmer from having to divvy up the system's behavior along the lines of its structure, endless new possibilities open up. A central possibility involves changes and updates. For example, this style should make it possible to conveniently remove pieces of behavior that we don't like and replace them with others, which is quite different from replacing a tangible part of the system with some new one. I would love to be able to reprogram the interactions that the web-based systems I work with force me to follow—not to mention reprogramming my annoying and unnecessarily complicated DVD. I can't change the way Amazon or B&H respond to what I do, for example, but I can surely change everything that has to do with the way *my* browser and *my* computer deal with these websites. And how better to do that than by simply canceling some pieces of interactive behavior and playing in new ones, using the very interface on which we interact, subject, of course, to my inability to change their behavior?

As to inheritance, my take is that its most interesting (and eventually useful) facet involves substituting objects in ways that preserve specific interobject behaviors. A good example involves replacing your secretary. You don't mind working with another secretary with different ways of doing things (and indeed with a possibly different set of things he or she can do). However, you want to ensure that the new person will be allowed to replace the old one only if certain behaviors are preserved. Here you would have the flexibility of detaching the structure (the substitutability of an object) from behavior (the maintained/inherited behaviors).

If hard-pressed to say in a nutshell what the play-in idea embodies, I would emphasize the fact that it is intended to constitute a natural and smooth computerization of how we'd cause some entity to behave the way we want. The programmer teaches and guides the computer to get to "know" about the system's intended behavior under development. This is done by working with—nay, playing with—the system itself or some soft version of it, and it should be done in the way that most naturally reflects how the programmer thinks about that behavior. It can contain any number and any combination of complicated modality-rich pieces of behavior, which can in turn involve many pieces of the system, intermixed with local behaviors, and they can be temporally short or lengthy. These pieces should be allowed to express actual operational instructions, as well as examples, guidelines, rules and constraints, and so on. Whatever is natural for the programmer, and can be conveyed to the system under development by an intuitive hands-on process, should be allowed.

Of course, this sounds exceedingly naïve, offering little more than the simple statement that almost anything goes. And that's easy to say in a section about how to program, but it generates a heavy debt, one that will have to be repaid when we talk about how to *run* those "programs."

THE DREAM, PART 2: RUNNING THE PROGRAM

So now we have to discuss what happens during and after play-in. Although play-in doesn't *seem* to require coding in some language, the play-in process itself is a language of sorts, and it's something formal-looking that the computer understands will have to be generated as its result. Here "understand" must mean, at the very least, "knows how to execute/run."

How can we do that? Well, since we haven't yet left dream mode, I can still answer in lofty words: I believe we can, and should, harness all the power of computing to do exactly that.

To start with, play-in must be worked out in such a way that, as behavior is being played in, the process is somehow recorded. It would have to be subjected to the required on-the-fly processing, including possibly speech and natural-language recognition, and then to formalization and logical capture. This would have the effect of

transforming the play-in sequence into a formal artifact, whose semantics captures the properties of, or the constraints on, the allowed traces of system behavior.

In fact, we can view play-in as a means for coaxing temporal specifications of behavior out of the user, and a natural medium in which to formalize these specifications would be some version of *temporal logic*. This would then be the "code" resulting from the piece of programming carried out. Learning theory and other AI techniques might very well be needed here to intelligently generalize examples into generic behaviors and to become smarter at understanding the more elaborate behavior that might be played in later. Thus, play-in will have to be formulated as a "language" amenable to this kind of analysis.

My firm conviction (and experience) is that even very liberal and informal notions of play-in will yield to such a formalization approach, as long as the programmer can get immediate feedback about how the playing in has been rendered. This must include making it possible for the programmer to observe both the generated formal version of the played behavior as well as its immediate effect on the played interface.

This kind of consistency and realizability checking, as well as the computing of resulting behavior, does not seem wildly impossible.

The main thing however, is this: At any point in the play-in programming process—and a special case of this is when the programming is over and we have the final system that must start operating—the programmer can ask to run the current version of the program. This really means that powerful and heavy computing would be employed transparently to execute the grand total of all the played-in pieces of behavior. We have termed this process *play-out*,[6] and it should be doable in interpreter-style mode (direct execution) or in compiler-style mode (synthesizing an executable). The programmer should be able to play with the play-out, so to speak, moving around among the possibilities and narrowing things down—all naturally and intuitively. Of course, if needed, the programmer should also be allowed to make changes in the formal rendition.

Again, this is easier said than done. What does it mean to execute, or play out, the grand total of the played-in behaviors? This is best explained by going back to our earlier metaphor.

Not only do I dream of *programming* computers the way we educate children, teach students, or make good citizens, but also of *running* those programs the way we expect that those people then go off and proceed to live their lives. Whenever we feel as if we have given the system enough instructions and guidelines, we set it free to start behaving. It can then do whatever it feels like, as long as it adheres to whatever was programmed into it during play-in. Whatever we told it that it *must* do, it will indeed do; whatever we said it is *not allowed* to do, it will never do; whatever we said it *might* do (for example, a nondeterministic or probabilistic choice among several possibilities), it will decide whether to do or not in the appropriate fashion, and so on.

In fact, if we ourselves choose to be fully and pedantically obedient (something that, interestingly, humans cannot really be expected to be but computers can …), that would be exactly how we would manage our lives. We all have our "books" of rules, containing all manner of instructions, regulations, guidelines, and laws relevant to our existence, the elements of which come in a variety of degrees of detail and explicitness. If we choose to adopt the good citizen stand, we will carry out the algorithm just described: We'll live any way we choose, as long as it is within the confines of those books.

Can we reliably transfer this procedure to the formal realm of computing? And if so, what kind of computational tools would doing so require?

We must somehow generate actual behaviors of the system that are consistent with the collection of played-in pieces of behavior; this is often called *realizability*. And recall that these pieces are multimodal and can contain constraints as well as operational instructions, and thus may limit, or even contradict, each other, and the entire approach will clearly be highly nondeterministic. (Is this the in silico version of free will?)

The word "consistent" here is crucial. I believe that we could develop powerful computational techniques and use them to verify the consistency of what was played in, to compute and lay out a plethora of traces of behavior that are consistent with all of that, and then to choose which of them to actually run/execute. These computations should be doable on the fly, so that the programmer can be warned that a behavior being played-in contradicts what has already been programmed and to provide immediate feedback. More elaborate versions of such computations that could yield more efficient execution instructions could be done offline, in a compile-like mode. This difference is not that important to the issue itself.

What is interesting is that this kind of consistency and realizability checking, as well as the computing of resulting behavior, does not seem wildly impossible. Consider the former: What we are really talking about is a kind of *verification* problem, except that, since here the hows and the whats are intermixed; verifying one against the other is really just checking the consistency or realizability of the compound "program." Similarly, computing legal behaviors of the system, if indeed it has any, is often called *synthesis*, or *temporal synthesis*.[10]

Both verification and the related notion of synthesis have been the subjects of extensive research efforts,

Some Evidence of Feasibility: Scenario-Based Programming

Having had the play-in bug in mind for a long time, and wanting to see whether a play-in approach to specifying reactivity was at all possible, it became clear that we needed a language that was intuitive enough for engineers and programmers to use, but which was not limited to specifying behavior per object. The whole idea of play-in calls for freedom in talking about the interaction between the system's parts.

The turning point came in the 1998 collaboration with Werner Damm, which resulted in the language of *live sequence charts* (LSCs).[1] This is a temporal visual formalism that extends classical message sequence charts (MSCs), or their UML variant, sequence diagrams, mainly by being *multimodal*, allowing existential and universal flavors both for the charts themselves and for the internal elements. (For the latter these flavors are called *hot* and *cold*.) LSCs were defined in the natural framework of object-oriented systems, and are also expressible in temporal logic. They make it possible to talk in operational terms about the interaction between the system and its environment and among the system's objects. We use the term *interobject* for this and use the term *intraobject* for the more conventional object-by-object specifications. The language allows specifying scenarios of what can and might happen (like those of MSCs), but in the case of LSCs also what must happen, what is not allowed to happen, and much more.

For the next several years, an extensive collaboration started with then-PhD student Rami Marelly. That work addressed several issues.[2] The first was to strengthen the original version of LSCs considerably, increasing its expressive power by adding several crucial features. The main ones were the notion of time (and a sort of real time), and a notion of genericity via variables and symbolic instances. Genericity allows using the play-in process to specify by-example scenarios, such as making a specific phone call using specific objects (the numeric keys) but where the result of playing it in will be a generic chart that refers to any such objects and therefore captures making any call.

The two major results of the work with Marelly were *play-in* and *play-out* for the full language of LSCs, and the construction of the Play-Engine tool that supports the two techniques.[2] For play-in, we use a GUI for the system's objects (whose internal local methods have to be given up front, separately), and the kinds of play-in processes allowed are in line with the structure and flow of an LSC. So we essentially play-in an LSC via the GUI by clicking for activation, right-clicking for method and action menu selection, and so on. We use icons on the tools' interface to select hot or cold, symbolic or not, and other possibilities for the semantics of what is being played in. As we play in, the system generates and displays the corresponding LSC on the fly, and its effect on the GUI is shown continuously. Using the Play-Engine, we can actually play in the behavior of a mobile phone just as the text of this article describes, but the porcess is more rigid.

As to play-out, the child/student/citizen algorithm is implemented as is. The Play-Engine keeps track of all live (= active), or potentially live, LSCs simultaneously, including multiple live copies of the same chart with different object or value instantiations. At each step, the algorithm figures out what actions are possible as next steps, taking into account the entire set of possibilities, rules and constraints, from all the charts. Hot things are always done, cold ones might be done, and forbidden ones are never done. When a contradiction occurs—for example, a clash between something that has to be done and something that must not—the system reports a violation and stops. As in play-in, play-out is carried out on the GUI, which responds and provides full visual feedback about the run, and the executing LSCs are also animated in the background.

There is something very declarative and nondeterministic about LSCs. The basic "naïve" play-out mechanism deals with the nondeterminism inherent in the language just as most software development tools that execute models deal with racing conditions: It simply chooses one of the possible next things to do and does it. Of course, this can lead to violations, which could have been avoided had another path been taken instead. It could also have been avoided had we been able to carry out full temporal synthesis, since if the specification is known to be consistent—that is, realizable—there is a guaranteed way to make progress, and play-out need never fail.

Since synthesis is still a rather futuristic possibility, we came up with *smart play-out*.[2] In this technique, which is the heart of former student Hillel Kugler's PhD thesis, the tool translates the problem of finding a nonviolating superstep—that is, a sequence of actions that the

both on their limitations—for example, the undecidability of certain general versions and identifying decidable and tractable subcases—and on their efficiency and practicality in actual usage.[11,12] In addition to these, it also seems clear that planning algorithms, theorem proving, inductive logical reasoning, heuristics, and probabilistic techniques will turn out to be crucial.

Another major issue that needs to be raised is distribution and final code. It is all very well to say that we will

system takes in response to an external event—into a verification problem and then employs model-checking to solve it. The system then promptly executes the resulting superstep in a way that is transparent to the user. This is quite possibly the first use of hardcore verification not to prove properties of programs but to run those programs.

While smart play-out is currently limited to a single superstep, it is not too difficult to see that (in principle) extending the idea to unlimited depth of the tree of supersteps would be tantamount to solving the consistency/realizability problem for LSCs and could lead to play-out strategies that fail only if there is no other possibility. The lesson to be learned from smart play-out, I think, is this: We know that verification techniques are becoming very good at proving that programs do what we want; let's now harness them to help get those programs to do what we want.

Following this basic work, for which we use the term *scenario-based programming*, several more advanced pieces of work were carried out. We have recently devised another algorithm for executing LSCs, *planned play-out*, which uses AI-style planning algorithms to do essentially the same as smart play-out. The usage of planning here is not surprising, as planning can be viewed as a special case of synthesis. One benefit of planned play-out is that we can use it to find more than one possible superstep. Also, taking advantage of the fact that planned play-out works in interpreter mode, we have also implemented an exploration mechanism that allows the user to navigate among possibilities during execution, trying things out, backtracking, and so on.

Both these non-naïve methods—smart and planned play-out—follow the original play-out mechanism in that they are interpreter-style approaches to execution. However, we have not yet applied either of them to the full LSC language, with time and symbolic instances being the main features that cause difficulties. In contrast to this, we have also exploited the similarities between aspect-oriented programming and the interobject nature of LSCs to build a compiler for a variant of LSCs, which translates them into AspectJ. This is more than the usual kind of compiler-style downward translation: The LSCs are compiled into Java, to which are added what we call *scenario aspects* to coordinate the simultaneous monitoring and direct execution of the compiled LSCs. We can then compile the generated Java code and link it to a separately implemented Java program to create a single executable application.

In other work, we have investigated the possibility of synthesizing state machines for the separate objects from the LSCs; we have proposed a way to incorporate a behavioral and object hierarchy into the language to help scenario-based programming scale up. We also have worked out the corresponding enriched play-out technique, and we have devised a distributed play-out protocol for a subset of the LSC language. On a somewhat different note, we have built a linking tool, called *InterPlay*, which programmers can use to mix interobject behavior given in LSCs with separate behavior given for some of the objects in an intraobject language, such as conventional code or statecharts. We have also been investigating how LSCs and the Play-Engine fare in some specific application areas, such as telecommunication systems, tactical simulators, biological modeling, and web services.

Nevertheless, while definitely relevant to the dream of liberating programming from its three restraining straightjackets, all this work is still partial and preliminary. There are many serious issues that need to be resolved even if we restrict ourselves to the scenario-based language of LSCs and the relatively modest versions of play-in and play-out that have already been worked out. We have not yet dealt with consistency, or realizability, except in the limited scope of a single superstep, nor have we paid much attention to the optimization of the various execution mechanisms. And determining how to scale scenario-based programming up to large, multilevel systems will require more than an adequate definition of hierarchical LSCs and distributed play-out.

References

1. W. Damm and D. Harel, "LSCs: Breathing Life into Message Sequence Charts," *Formal Methods in System Design*, vol. 19, no. 1, 2001, pp. 45-80.
2. D. Harel and R. Marelly, *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*, Springer-Verlag, 2003.

use all kinds of techniques to compute the "good citizen" live-by-the-rules idea of play-out, but there is a nagging feeling that a naïve approach to this—even if computationally very powerful—would have to somehow generate an overall controller or scheduler for the entire system. For real-world systems (again, we need only consider web applications as an alarming example), by the end of the day we are often required to have actual code running on separate machines, and possibly in separate locations, working together to constitute the running system.

It's nice to dream of specifying and executing behavior in a way that is orthogonal to the system's breakup into parts, but the final system implementation very often must adhere to the boundaries of those parts, and quite possibly this would include at least parts of the system's behavior too. So we will probably have to find ways to *distribute* the intuitive played-in behavior, breaking it up into pieces that the system's various parts can deal with. This kind of distributed play-out is highly nontrivial and calls for a distributed variant of the synthesis problem, which happens to be even harder and is not as well understood.[12] It will probably require the development of ever more powerful techniques and ideas connecting distributed and parallel computing with logic and verification-like methods.

ON SCENARIO-BASED PROGRAMMING

The material in the "Some Evidence of Feasibility: Scenario-Based Programming" sidebar is not imaginative—it is real work that has been done over the past nine years jointly with a group of greatly talented colleagues and students. When compared to the grandiose spirit of the previous comments, however, it is partial, fragmented, and rather narrow. For example, it is restricted to programming the reactive and interactive dynamics of sets of objects, and it doesn't attempt to deal with any algorithmic or data-intensive types of programming. Thus, even the potential scope of the dream discussed here does not become clear from it. Nevertheless, I maintain that it provides some evidence that it might be worth thinking more seriously about liberating programming along the lines I have discussed here.

In particular, using the language of LSCs and the Play-Engine discussed briefly in the sidebar, we can actually play in behavior similar to the cell phone example described earlier, albeit far more rigidly, and we can then play out the set of behaviors according to the "good citizen" algorithm. And all this is done in an interobject, rather than intraobject, fashion. In some of our more recent work, we have used well-known techniques from verification and AI, and this also adds to the feeling of feasibility that it raises.[8]

The bottom line is this: I believe there is no reason why we shouldn't make great efforts to bring widely researched and deeply worked-out ideas in computer science to bear upon the most basic and profound activity that involves computers, namely, programming them and running the resulting programs. Once liberated, programmers will probably have new kinds of work to do, possibly including the need to set up specialized features of the new sophisticated computational tools that would be running in the background.

There is obviously a great deal more to programming than specifying the reactive and interactive give and take of objects. I can imagine some ways in which the ideas described (or hallucinated about) here might be extended to other kinds of programming, involving other kinds of entities, such as classical algorithmics, data structures, and databases. But I'd be the first to admit that there are many more things that are relevant to all of this, about which I don't even know enough to dream of, let alone to imagine how to do. Also, I am not saying that any of this is easy, or even that it is clear that it can be done. Such is the nature of dreams.

On the other hand, dreaming and sharing the dreams with others has never been a mortal sin. ■

Acknowledgments

I would like to express my deepest thanks to the many dedicated and talented people whom I have had the pleasure of working with on scenario-based programming over the past few years. Special thanks go to Werner Damm, Rami Marelly, and Hillel Kugler, without whose lengthy collaboration even the dreaming would have been impossible. The other members and ex-members of my group who were actively involved in developing the ideas described in the sidebar include Shahar Maoz, Yoram Atir, Dan Barak, Asaf Kleinbort, Ron Merom, Ouri Poupko, and Itai Segall. Thanks also to Shahar Maoz and Moshe Vardi for comments on the manuscript. This article was written during a visit to the School of Informatics at the University of Edinburgh and was supported by a grant from the EPSRC. The research described in the sidebar was supported in part by the John von Neumann Center for the Development of Reactive Systems at the Weizmann Institute of Science, by a Minerva grant, by a collaborative NIH grant, and by the Israel Science Foundation.

References

1. D. Harel, "From Play-In Scenarios to Code: An Achievable Dream," *Computer*, Jan. 2001, pp. 53-60. Also published in *Proc. Fundamental Approaches to Software Engineering* (FASE 00), LNCS 1783, Springer-Verlag, 2000, pp. 22-34.
2. J. Backus, "Can Programming Be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs," *Comm. ACM*, vol. 21, no. 8, 1978, pp. 613-641.
3. M. Zloof, "Query-by-Example: A Data Base Language," *IBM Systems J.*, vol. 16, 1977, pp. 324-343.
4. K. Apt, *From Logic Programming to PROLOG*, Prentice Hall, 1996.
5. C. Simonyi, *The Death of Computer Languages, The Birth of Intentional Programming*, tech. report MSR-TR-95-52, Microsoft Research, 1995.
6. K. Marriott and P.J. Stuckey, *Programming with Constraints: An Introduction*, MIT Press, 1998.

7. R.E. Filman et al., *Aspect-Oriented Software Development*, Addison-Wesley, 2004.
8. D. Harel and R. Marelly, *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine*, Springer-Verlag, 2003.
9. D. Harel and A. Pnueli, "On the Development of Reactive Systems," *Logics and Models of Concurrent Systems*, K.R. Apt, ed., NATO ASI Series, vol. F-13, Springer-Verlag, 1985, pp. 477-498.
10. A. Pnueli and R. Rosner, "On the Synthesis of a Reactive Module," *Proc. 16th ACM Symp. Principles of Programming Languages*, ACM Press, 1989, pp. 179-190.
11. E.M. Clarke, O. Grumberg, and D.A. Peled, *Model Checking*, MIT Press, 2000.
12. M.Y. Vardi, "From Verification to Synthesis," presentation, 2006 (in PostScript); www.cs.rice.edu/~vardi/papers/fmco06.ps.gz.

David Harel *is the William Sussman Professor in the Department of Computer Science and Applied Mathematics at the Weizmann Institute of Science. Harel invented statecharts, coinvented live-sequence charts, and was a member of the team that designed Statemate and Rhapsody. He also codeveloped the play-in/play-out approach to scenario-based programming and the Play-Engine. Recently, he has also been working on odor communication and biological modeling. He is a fellow of the IEEE, the AAAS, and the ACM. Contact him at dharel@weizmann.ac.il.*

An Assessment of Integrated Digital Cellular Automata Architectures

Victor Zhirnov and Ralph Cavin, Semiconductor Research Corporation
Greg Leeming, Microelectronics Advanced Research Corporation
Kosmas Galatsis, FCRP Center on Functional Engineered Nano Architectonics, UCLA

Future nanoscale technology might drive a migration to different information-processing and computing approaches. One such possibility is the class of digital cellular automata. The recent emergence of multicore architectures, driven by semiconductor technology constraints, motivates the investigation of cellular automata architectures as information-processing alternatives.

With the recent introduction of dual-core processors, the wide-scale transition to parallel processing might have finally begun. How will multicore processing evolve and what will be the dominant computer architecture of the future? As technology reaches the limits of CMOS and beyond, the physical realities of computing hardware might dictate the answer to these questions.

The integration level for nanoscale electronic devices could eventually be in the range of 10^{10} to 10^{11} devices per square centimeter.[1] At this level long interconnects represent a significant challenge to operation (energy consumption), design, and manufacturing (irregular arrays of interconnects with arbitrary connections). Also, nanoscale elements are likely to suffer from significantly higher failure rates than their contemporary counterparts. In addition, low-energy operation requirements and small transistor dimensions are likely to result in higher thermal and quantum error rates. Moreover, the problem of designing complex irregular structures at these density levels is becoming increasingly untenable.

Given these realities, future nanoscale technology may drive a migration to different information-processing and computing approaches. One such possibility is the class of digital cellular automata. The recent emergence of multicore architectures, driven by semiconductor technology constraints, motivates the investigation of cellular automata architectures as information-processing alternatives.

This possible migration to new computing architectures is theoretically predictable. Gianfranco Bilardi and Franco Preparata[2] elegantly argued that, as computing technology approaches the physical limits of scaling (as dictated by the speed of light), ultimate device size reduction limits, and realizable fan-out and fan-in device constraints, it will naturally drive architectures of practical interest toward regular arrays of locally connected computational elements.

CELLULAR AUTOMATA ARCHITECTURES

Cellular automata architectures are possible alternatives to so-called von Neumann architectures. Historically, von Neumann proposed both approaches in the early 1950s.[3]

As Figure 1a shows, the traditional von Neumann or sequential architecture refers to a computer that consists of a processing unit (PU) and memory (M) unit. The memory stores both instructions and data, and the sequence of logical operations is based on flags and control lines. Such a system of *localized* memory and logic units can implement a *general-purpose computer.*

On the other hand, as Figure 1b shows, cellular automata are composed of identical cells with the same connec-

Published by the IEEE Computer Society
0018-9162/08/$25.00 © 2008 IEEE

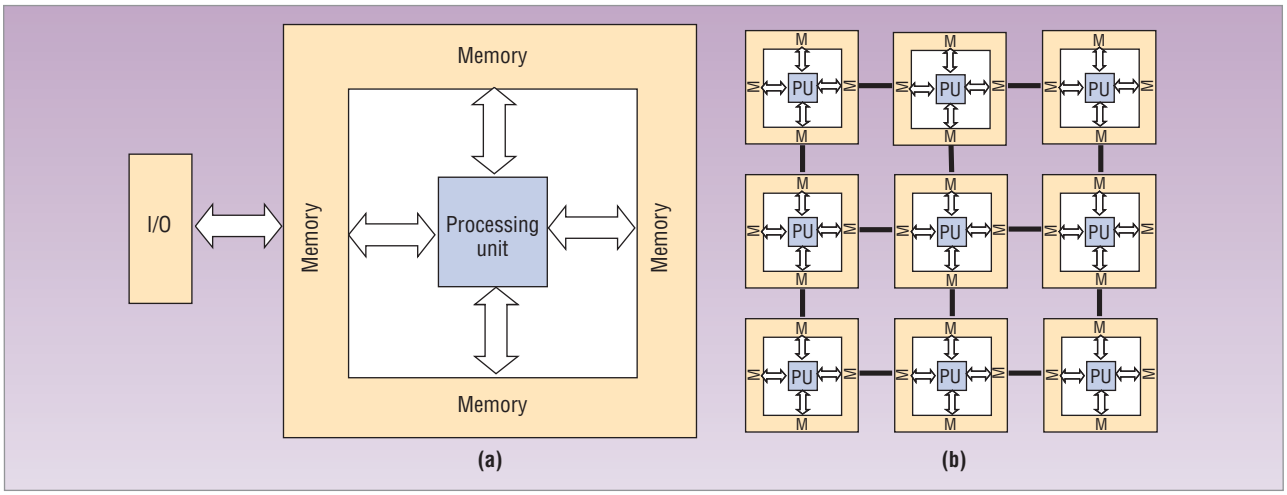


Figure 1. Two generic computer architectures: (a) von Neumann sequential architecture, and (b) cellular array architecture.

tion structure to neighbors. Only a cell's state can distinguish it from its neighbors. Each cell's state is a certain function (a "rule") of the states of neighboring cells.

CELLULAR AUTOMATA FOR MAINSTREAM COMPUTING

In our endeavor to explore whether cellular automata could be a viable mainstream computing implementation, we seek to answer the following key questions:

- What is the minimal required internal complexity of cells in cellular automata that can support general computation?
- How many cells are required for digital cellular automata architectures to become as computationally efficient as von Neumann architectures for general computation?
- Can beyond-CMOS devices provide a substantive computational benefit to cellular automata architectures?
- Is interconnect complexity reduced in a cellular automata architecture?
- Do feasible methods exist to tolerate hard and soft errors in cellular automata?
- Do cellular automata architectures offer power, reliability, design, or manufacturability benefits?

Our analysis will be limited to the case of "digital" cellular automata and is independent of specific hardware implementations.

Minimal complexity cells for cellular automata and cellular arrays

Each cell in a cellular automaton contains a certain number of discrete elements—transistors, resistors, diodes, and so on. The cell's internal complexity—the number of discrete elements—defines the cell function and therefore the operation of the cellular automaton. As von Neumann put it,[3]

> … if one constructs the automaton (A) correctly, then any additional requirements about the automaton can be handled by sufficiently elaborated instructions. This is only true if A is sufficiently complicated, if it has reached a certain minimum of complexity.

Essentially, a cellular automata cell must surpass a certain internal complexity threshold, referred to here as "von Neumann's threshold," if it is to perform arbitrarily complex tasks by virtue of elaborate software instructions. For example, let's consider a 1-bit general-purpose processor, which contains an arithmetic logic unit and sufficient memory. We can demonstrate that, with a minimum set of operations, a 1-bit ALU will contain about 98 discrete elements. The addition of essential memory requirements results in a minimum cell complexity on the order of about 150 to 200 elements. This is consistent with von Neumann's estimate, suggesting that the minimum core complexity required to implement general-purpose computing is on the order of a few hundred binary switches.[4]

Cellular automata might not require the full functionality of a general-purpose processor for each cell. In this case, we could implement less-complex cells. In cellular automata, each cell can be in an "alive" or "dead" state, depending on the state of its neighbors.

Consider a two-dimensional array, having eight adjacent cells: N, NE, E, SE, S, SW, W, and NW. The implemented rule must consider $2^8 = 256$ possible combinations of adjacent cell states. Typically one selected rule is applied uniformly to all the cells. We can estimate the minimal cell complexity in a "maximal-rule" cellular automaton as follows:

- The cell senses the independent states of eight neighbors, requiring at least eight elements.
- The cell sends information about its own state to its neighbors, requiring at least one element.

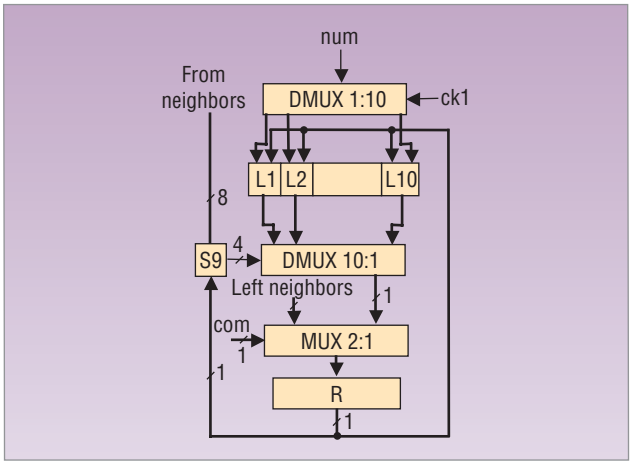


Figure 2. Schematic diagram of the cell in a totalistic cellular automaton. (Adapted from M. Dascalu and E. Franti.[5])

- The cell responds to eight independent inputs according to a certain rule using an 8 to 256 decoder, requiring approximately 2,048 elements.
- Cells can be programmed to execute one of 256 rules, requiring at least 256 elements.

We obtain the minimal cell complexity by adding all contributions, and it exceeds 2,300 elements for a 256-rule cellular automaton. This complexity is directly due to the high number of local rules that the cells might need to execute. (Note that the number of elements of these cells is the same as in a basic microprocessor such as Intel's 4004.) High cell complexity and size is acknowledged as the main problem of cellular automata implementations in their generalized form.[5] To reduce the size of cellular automata, the number of rules must be decreased even though this compromises computational efficiency and universality.[5]

For a fixed-rule machine, the 256 discrete elements required for rule programming can be eliminated. Also, one combinational circuit, for example, an AND requiring eight elements at a minimum, can be substituted for the 8 to 256 decoder. For the limiting case of a fixed single-rule cellular automaton with a five-cell neighborhood, the minimal complexity of a fixed-rule cell is greater than $2^4 + 1 = 17$. However, it should be noted that such a machine has limited practical applicability.

One approach is to use a reduced-rule-set cellular automaton implementing the *totalistic* binary functions whose output value depends only on the sum of all of their input variables, not on the value of each input variable.[5] There are several totalistic rules of potential interest, for example, the *majority function*, the *exclusive or*, and the *Game of Life*. Figure 2 shows a schematic diagram of the cell in a totalistic cellular automaton. It consists of a multiplexer 2:1 (MUX 2:1), a demultiplexer 1:10 (DMUX 1:10), a multiplexer 10:1 (MUX 10:1), 10 addressable latches, the S9 circuit (a 9-bits sum), and a 1-bit register.

Hence, we can conclude that a minimum cell will consist of at least several hundred discrete elements, comparable in complexity to the basic single-bit general-purpose computing element.

CELLULAR AUTOMATA IN 2020

A 2006 high-performance MPU contains about 400 million transistors per square centimeter and might approach 7 billion transistors by 2020.[1] If a minimum of approximately 200 components is required per cellular automata cell, future semiconductor processes could conceivably deliver tens of millions of minimal-complexity cells per square centimeter.

How would the performance of a cellular automaton containing this number of cells compare with that of a contemporary 2020 von Neumann architected MPU? Clearly, if the cellular array offers superior performance, we should expect substantive changes in microprocessor design in the future.

We are unaware of an authoritative answer to this question. Most of the literature discussing such comparisons argues that both Turing machines and cellular automata can perform universal computation, where the universal computer has the following capabilities[6]:

- a means of communicating to the outside world with the purpose of receiving input and producing output at any time during computation;
- the ability to perform all elementary arithmetic and logical operations;
- a program made up of basic input, output, arithmetic, and logical operations; and
- an unlimited memory in which programs, the input, intermediate results, and the output can be restored and retrieved.

Several classes of cellular automata have been shown to satisfy the properties of a universal computer, usually by showing equivalence to the Turing machine. The Turing machine has infinite memory by construction whereas universal cellular automata are infinite in extent. Of course, realizable general-purpose computers do not possess this infinite storage capability, yet they can do useful work.

Several specialized algorithms have been demonstrated, for example in fluidics and pattern recognition, where, due to the local nature of the required computation, cellular automata demonstrate significant computational advantage. However, an interesting question is: What hardware complexity does a finite cellular automaton require to obtain equivalence with a simple finite von Neumann computer, for example, a one-bit microprocessor with limited memory? Since it should be fairly straightforward to characterize a one-bit microprocessor, we think this would make an interesting comparison basis for finite cellular automata.

Beyond CMOS nanoelectronic implementations of cellular automata

In digital cellular automata, as in all digital circuits, a binary switch is the fundamental building block. The research community has proposed several alternatives for the CMOS binary switch. All possible realizations of electron-based binary switches face the same limits and tradeoffs for size, speed, and power dissipation.[7,8] The 2005 ITRS reviewed these alternative devices and concluded that none appear to be sufficiently mature to offer competition to the CMOS switch.[1,8]

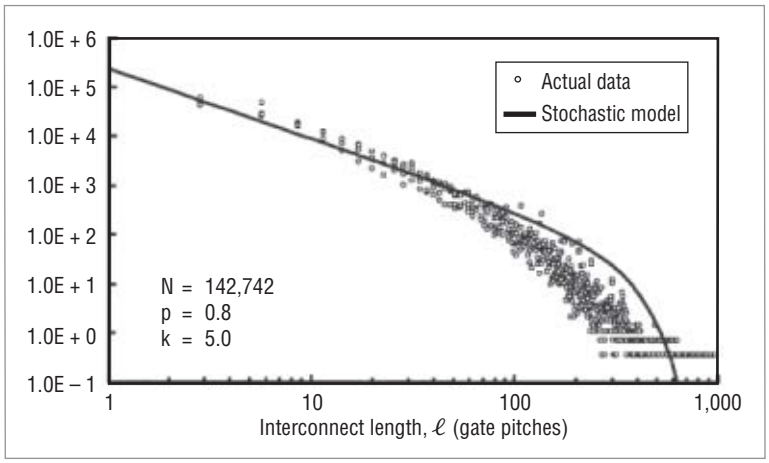


Figure 3. Typical interconnect-length distribution for microprocessors. (Adapted from J.A. Davis, R. Venkatesan, and J.D. Meindl.[9])

Interconnects in CA architectures

Figure 3 shows the interconnect length distribution for general-purpose processors.[9] Jeffrey Davis and colleagues offered a rigorous derivation of a complete wire-length distribution for on-chip arbitrary logic networks based on Rent's rule.[9] The theoretical distribution is very close to the empirical relationship shown in Figure 3. In turn, other researchers have shown that a sufficient condition for the appearance of the power-law form of Rent's rule is the statistical homogeneity of gate placement.[10]

We can use the interconnect-length distribution of Figure 3 to calculate an average wire length. In general, the average wire length L (n) is a function of the transistor density, n. Table 1 shows the average wire length L (n) for different n and gate length L_g (calculated using Davis's approach).[9]

We now consider the wire-length distribution in cellular automata architectures. Since we have shown that a cellular automaton cell is likely to have the complexity of a low-end microprocessor, we will assume that each cell requires an interconnect-length distribution equivalent to that of a general-purpose processor and that the interconnects between neighboring cells are minimal—that is, a single conductor. For the four-neighbor case, this results in one interconnect per cell. The assumption of one interconnect between cells is most favorable for cellular automata since this will provide a lower bound on interconnect power dissipation estimates. The intercell interconnect's minimum length is approximately given by the average separation between two neighboring transistors in an integrated circuit. For a statistically homogeneous transistor placement, an average separation between two neighboring transistors $L_{t\text{-}t}$ is approximately 10 L_g.

To estimate the average interconnect length in a cellular automata architecture, consider a chip with the total number of transistors N. Let each cell consist of M transistors so that the number of cells is $K = N/M$. If each cell possesses a distribution similar to a general-purpose processor, the interconnect-length distribution inside the cell is typically represented by the distribution shown in Figure 3.

Outside the cells (elementary processors), there are only local short interconnects with length about $L_{t\text{-}t}$ (2) between neighboring cells. The number of intercell interconnects is K (one wire per cell). If we consider a maximum-density chip (N~10^{12} in 1 cm^2) organized in a single-core ($K = 1$) and a cellular architecture, the average wire length for a cellular automata architecture is less than that for a single-core architecture, as Table 1 shows.

Whereas the average wire length for a single-core architecture is 4.1, a cellular architecture with a minimum cell complexity, $M_{\min}$~100, $(L_1/L_K)_{\max}$, is only 2.64. This suggests that, in principle, a cellular implementation can decrease the energy dissipation from interconnects. Note that for larger cell complexity, this ratio naturally would decrease. It should also be noted that additional interconnections might be required for initialization and control lines to program and operate a useful cellular automaton. The presence of such global interconnections might downgrade the potential benefits of cellular automata architectures.

Fault tolerance of cellular automata

A major challenge associated with fine-grained architectures is the foreseeable degree of nanocomponent defects and faults. Several researchers have addressed the fault tolerance of cellular automata.[11-14] Youichi Nishio and Hidenosuki Kobuchi described one early attempt to construct a fault-tolerant cellular automaton.[11] In this model, the maximum number of errors that can be tolerated—that is, corrected—is one in 19 cells. To enable such a level of fault tolerance, each cell needs to be connected to at least 49 neighboring cells—in other words, the very principle of local connectivity is broken. Other researchers have explored fault tolerance in infinite cellular automata[12,14] and have shown that fault-tolerant computation by an infinite cellular automaton is possible in principle.

Table 1. Average wire length in a cellular array chip for different numbers of transistors in a 1 cm² chip *N* and the number of transistors in each cell *M* (the number of cells *K* = *N*/*M*). The average wire length is normalized to the gate length. L_{cell} (*M*) is the average wire length inside the cell, L_K is the average wire length across the chip of *K* cells, and L_i is the average wire length of a single-core architecture (*K* = 1).

N	M	K	L_{cell} (M)/L_g	L_i/L_K
10^{12}	10^2	10^{10}	4.1	2.64
10^{12}	10^3	10^9	5.3	2.09
10^{12}	10^4	10^8	6.4	1.72
10^{12}	10^5	10^7	7.5	1.48
10^{12}	10^6	10^6	8.3	1.33
10^{12}	10^7	10^5	9.1	1.22
10^{12}	10^8	10^4	9.7	1.14
10^{12}	10^9	10^3	10.2	1.09
10^{12}	10^{10}	10^2	10.5	1.05
10^{12}	10^{11}	10	10.8	1.02
10^{12}	10^{12}	1	11.1	1.00

However, the results obtained for infinite media might not be relevant for realizable architectures.

Ferdinand Peper and colleagues[14] constructed an asynchronous cellular automaton based on delay-insensitive circuits that was asymptotically fault-tolerant to arbitrary errors in up to one-third of the bits representing the information in the cellular array.

In a good example of the study of robustness in cellular automata that was based on the Game of Life, a classic cellular automaton computational model, the authors studied the time evolution of the model's state at different temperatures.[15] They found that there is a critical temperature at which a given pattern decays and that these decay temperatures are different for different patterns. If the ratio of the "cell local energy"—for example, the energy needed to change the cell state—to the thermal energy is larger than 4.25, all major patterns survive without losing their shape, but for lower ratios, the patterns degrade due to the accumulation of errors. More studies of the time evolution of the patterns in a cellular array at finite temperatures are needed, especially in the application to implementation of specific logic operations.

Power, reliability, design, and manufacturability benefits

The emergence of cellular architectures will also depend on their power consumption, reliability, design complexity, and manufacturability characteristics relative to their von Neumann architecture counterparts. Since cellular architectures have not been developed to the same degree as today's von Neumann architectures, we can only surmise which features might prove to be advantageous if these architectures do make it into the mainstream.

The thermal performance challenges associated with VLSI design are well known. How will the power requirements and thermal performance of cellular automata compare with von Neumann designs? As suggested, the same devices used to implement von Neumann architectures will likely dominate cellular automata hardware implementations. Therefore, at the component level, cellular automata power requirements and thermal performance are unlikely to be significantly different from von Neumann implementations. However, digital cellular automata might realize a power and thermal advantage at the macro level. The extreme regularity of these architectures should minimize the likelihood of thermal hot spots. Power consumption should be relatively uniform across the chip, resulting in simpler power distribution and heat removal designs.

Reliability is predicted to become increasingly challenging as on-chip components shrink in accordance with Moore's law. Future components are predicted to suffer from higher variability as well as significantly higher failure rates. Cellular automata architectures should make the task of compensating for these future device characteristics easier.

The solution to component variability is likely to be the incorporation of significant compensation circuitry. In a cellular automaton, researchers need to design this specialty circuitry for only a single cell, which will then be duplicated across the entire array. System reliability issues are likely to be addressed through the application of redundancy. The extreme regularity of cellular automata naturally lends itself to redundant designs and should also be advantageous from a manufacturability standpoint.

Researchers have shown that a small cell library of standardized logic blocks (bricks) is sufficient for an efficient implementation of any microchip design.[16] The resultant regular designs lend themselves well to nanoscale manufacturing, relieving the increasing stress on lithography techniques as feature dimensions shrink. Hence, the inherent regularity of cellular automata should be advantageous from a manufacturability standpoint.

We began by seeking to answer six questions about the role of digital cellular automata architectures in future computational architectures, given the rapid advance of VLSI technology. Following in the spirit of the 2006 Focus Center Research Program Workshop on Computation in Nanoscale Dynamical Systems,[17] our approach has been to review the research literature and, where possible, to make esti-

mates of the minimal hardware complexity for minimal cellular automata and von Neumann elements.

We estimate that a flexible-rule cellular automaton would require 2,300 elements for each cell, surprisingly similar to the number of elements in a basic microprocessor such as the Intel 4004. But we could not determine how many cells a digital cellular automata architecture requires to become as computationally efficient as single von Neumann architectures for general computation. We have shown that claims of computational universality for cellular automata are usually based on arrays of infinite extent and have suggested that a cellular automata implementation of a one-bit microprocessor might shed some light on the resolution of this question on general computation.

On the downside, we have not been able to identify non-CMOS technologies that provide a performance or implementation advantage relative to CMOS implementations of cellular automata architectures. For the foreseeable future, we believe cellular automata will be implemented in CMOS technology. Other technologies do not appear to be mature enough to warrant consideration at this time.

On the upside, our analyses indicate that, relative to a single-core von Neumann uniprocessor, cellular automata architectures enjoy a mild advantage in the average length of interconnects. For example, for a minimum cell complexity, M_{min}~100, and assuming one billion cells, the cellular architecture implementation's average interconnect length is half that of a uniprocessor architecture. As such, we believe this could translate into energy savings in cellular automata interconnect systems.

While our analysis of tolerance for hard and soft errors in digital cellular automata architectures shows no difference between that in a von Neumann architecture, we believe that they might indeed offer benefits in these areas, primarily due to the innate regularity of the structures. Moreover, assuming uniform cell activity, cellular automata architectures might generate heat more uniformly on the chip and therefore ease heat management challenges.

Semiconductor technology trends favor the realization of regular, locally connected structures, and digital cellular automata conform well to this trend. However, if digital cellular automata are to have an impact on information-processing technology, it is important to demonstrate the capability to address classes of applications of general interest and importance.

In this light, we acknowledge focusing primarily on the hardware issues related to digital cellular automata. When we consider the use of these systems to implement computation for general applications, a vexing set of software challenges arise. For one, a compiler for cellular automata would need to be constructed to set the element rule schedule to implement the prescribed computation, and we are aware of little work in this area.

Acknowledgments

We acknowledge the contributions of Rick Kiehl at the University of Minnesota, Kovas Boguta at Wolfram, and Robert Colwell.

References

1. Semiconductor Industry Assoc., *International Technology Roadmap for Semiconductors, 2005 Edition*; www.itrs.net/reports.html.
2. G. Bilardi and F.P. Preparata, "Horizons of Parallel Computation," *J. Parallel and Distributed Computing*, vol. 27, no. 2, 1995, pp. 172-182.
3. J. von Neuman, *Theory of Self-Reproducing Automata*, Univ. of Illinois Press, 1966.
4. J. von Neumann, *The Computer and the Brain*, Yale Univ. Press, 1959.
5. M. Dascalu and E. Franti, "Implementation of Totalistic Cellular Automata," *Proc. CAS 2000*, IEEE Press, 2000, pp. 273-276.
6. S.G. Akl, "The Myth of Universal Computation," *Parallel Numerics*, R. Trobec et al., eds., Part 2, Systems and Simulation, Univ. of Salzburg, Austria, and Jozef Stefan Inst., Slovenia, 2005, pp. 211-236.
7. V.V. Zhirnov et al., "Limits to Binary Logic Switch Scaling—A Gedanken Model," *Proc. IEEE*, vol. 91, 2003, pp. 1934-1939.
8. V.V. Zhirnov et al., "Emerging Research Logic Devices," *IEEE Circuits & Devices Magazine*, vol. 21, 2005, pp. 37-46.
9. J.A. Davis, R. Venkatesan, and J.D. Meindl, "Stochastic Multilevel Interconnect Modeling and Optimization," *Interconnect Technology and Design for Gigascale Integration*, J.A. Davis and J.D. Meindl, eds., Kluwer Academic Publishers, 2003, pp. 219-262.
10. P. Christie and D. Stroobandt, "The Interpretation and Application of Rent's Rule," *IEEE Trans. Very Large-Scale Integration Systems*, vol. 8, 2000, pp. 639-648.
11. H. Nishio and Y. Kobuchi, "Fault-Tolerant Cellular Spaces," *J. Computer and System Sciences*, vol. 11, 1975, pp. 150-170.
12. P. Gacs, G. Kurdyumov, and L. Levin, "One-Dimensional Homogeneous Media Dissolving Finite Islands," *Problems of Information Transmission*, vol. 14, 1978, pp. 92-96.
13. P. Gacs, "Reliable Computation with Cellular Automata," *J. Computer System Science*, vol. 32, 1986, pp. 15-78.
14. F. Peper et al., "Fault-Tolerance in Nanocomputers: A Cellular Array Approach," *IEEE Trans. Nanotechnology*, vol. 3, 2004, pp. 187-201.
15. S. Adachi, F. Peper, and L. Jia, "The Game of Life at Finite Temperature," *Physica D*, vol. 198, 2004, pp. 182-196.
16. V.R.V. Kheterpal et al., "Design Methodology for IC Manufacturability Based on Regular Logic-Bricks," *Proc. 42nd ACM/IEEE Design Automation Conf.* (DAC 2005), IEEE Press, 2005, pp. 353-358.
17. FCRP Workshop on Computation in Nanoscale Dynamical Systems, 19-20 Jan. 2006; www.fena.org/downloads_public/CNDS%20Workshop%20051115.pdf.

Victor Zhirnov, *a program manager at the Semiconductor Research Corporation, is responsible for research in the area of emerging nanoelectronic devices. His research interests include properties of materials at nanoscale, fundamental limits of heat removal, and nanoelectronics. Zhirnov received a PhD in physics from the Institute of Physics and Technology, Moscow. He is a member of the editorial team for the chapter on Emerging Research Devices of the* International Technology Roadmap for Semiconductors. *Contact him at victor.zhirnov@src.org.*

Ralph Cavin *is chief scientist for the Semiconductor Research Corporation. His research interests are in semiconductor technologies, information-processing architectures, and computer-aided design. He received a PhD in electrical engineering from Auburn University. He is a Life Fellow of the IEEE. Contact him at Ralph.cavin@src.org.*

Greg Leeming *is an Intel assignee to the Microelectronics Advanced Research Corporation, where he is the program manager for the Focus Center Research Program. His research interests are in novel computer architectures and information processing. Leeming received an MS in electrical engineering from Brown University. He is a member of the IEEE. Contact him at greg.p.leeming@intel.com.*

Kosmas Galatsis *is the executive director of the FCRP, FENA, and WIN Centers at the University of California, Los Angeles. His research interests include nanodevices, nanopatterning, and nanoarchitectures. Galatsis received a PhD in electrical engineering from the Royal Melbourne Institute of Technology in Australia. He is a member of the IEEE. Contact him at kos@ee.ucla.edu.*

Toward a Competitive Pool-Playing Robot

Michael Greenspan, Joseph Lam, Marc Godard, Imran Zaidi, and Sam Jordan, Queen's University
Will Leckie, Nortel
Ken Anderson, Larus Technologies
Donna Dupuis, University of British Columbia

Deep Green is a vision-based, intelligent robotic system that currently shoots pool at a better-than-amateur level, with the ultimate goal of challenging a proficient human opponent at a championship level.

From behind a closed door in a university campus hallway comes the distinctive clacking sound of a pool game in progress. This isn't a student lounge, but rather a laboratory where we're developing a vision-based, intelligent robotic system to play competitive pool. Named Deep Green, the system currently shoots at a better-than-amateur level, and our goal is to advance the system to be able to challenge a proficient human opponent, ultimately at a championship level.

Pool—by which we loosely refer to all cue sports, including billiards, carom, and snooker—is somewhat misunderstood, more likely to evoke images of shifty characters in smoky bars than advanced robotics. It evolved in the royal courts of medieval Europe as an indoor version of croquet. Today, pool is enjoying a resurgence of popularity worldwide. Variations are played in almost every country, and pool was recognized as a demonstration sport in the 1998 Nagano Olympics. According to a 2005 survey,[1] more than 35 million people played pool that year in the US alone, and pool ranked as the eighth most popular participation sport, just after cycling and fishing.

The first attempt to automate pool was the Snooker Machine developed at the University of Bristol in the late 1980s,[2] which culminated with a televised game on BBC's science program *QED*.[3] Since then, researchers have developed a number of pool-playing robotic systems[4-6] as well as a training system that has a computer vision component but doesn't involve robotic actuation.[7,8]

DEEP GREEN

As Figure 1 shows, Deep Green is centered on a 3-degree-of-freedom (DOF) industrial gantry robot, which is mounted to the ceiling to avoid impeding human access to the table. A digital camera, the *global vision system* (GVS), is attached to the ceiling aiming down toward the table, accompanied by an array of directional lights. Attached to the gantry's vertical post is a 3-DOF spherical robotic wrist that, combined with the gantry's linear motion, affords the robot complete reachability over the workspace.

The *end-effector,* illustrated in Figure 2, includes two distinct cue devices, one based on a linear electromagnetic motor and the other actuated pneumatically. The electromagnetic cue can be finely controlled to strike up to a velocity of 3 meters per second, which is sufficient for normal play, whereas the pneumatic cue is used solely for power breaks and strikes at 12 m/s. A small eye-in-hand camera, the *local vision system* (LVS), is also attached to the end-effector, as is a pick-and-place vacuum tool for ball-in-hand conditions and automatic racking.

The table itself is a standard 4-foot × 8-foot coin-operated pool table, and all devices are connected to a single

Published by the IEEE Computer Society

PC. While the system has been designed to play the popular game of 8 Ball, with slight modifications it could play any other variation of pool. Figure 3 shows a number of example shots.

ROBOTICS

Rather than build our own hardware, we based Deep Green on standard commercially available, albeit customized, components. This makes the system relatively inexpensive and quick to deploy, and it allowed us to focus our effort on the computational challenges.

Camera calibration

The system's robotic aspects rely primarily on computer vision. Before using the cameras, we had to calibrate them so that they could accurately determine the ball locations within the table's metric coordinate reference frame. Using standard techniques, we determined the cameras' intrinsic parameters, including factors to correct for the radial distortion inherent to optical systems. It was also necessary to rectify the table plane to compensate for perspective distortions that result from the GVS retinal plane not being aligned exactly parallel to the table surface, which is difficult to achieve manually to the desired accuracy.

The retinal plane and the table are related by a transformation known as a *homography,* a mapping between two planes. The standard technique for determining a homography involves extracting a minimum of four corresponding point locations between a planar pattern and its image. This technique is awkward to apply in Deep Green as the pattern must be large (the table's size) as well as very flat and accurate.

Alternatively, we exploit an invariant property of the projective space that uses a simple target comprising perpendicular lines, such as a large carpenter's square. This technique lets us integrate measurements taken at various positions on the table into a single homography, which we estimate up to an affinity. With a few additional simple measurements, we can then recover the remaining rotation and scale parameters that map the image pixels to metric locations on the table surface.

Ball localization and identification

At runtime, Deep Green acquires a GVS image when the balls come to rest and unwarps it to remove the radial and perspective distortions. It then compares this image with a set of statistics—pixel means and variances—acquired from a set of approximately 30 background images of the table, without any balls present. For each pixel, if the difference between the foreground and background pixel values exceeds some threshold value of the background standard deviation, the system judges that pixel to be foreground, that is, possibly a ball.

Figure 1. Deep Green robotic pool-playing system. The system is centered on a 3-degree-of-freedom gantry robot mounted to the ceiling to avoid impeding human access to the table.

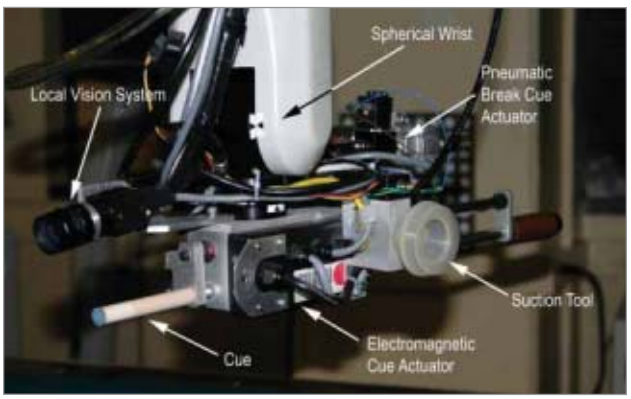


Figure 2. End-effector components.

Because this filter passes significant noise, the system applies a connected-components algorithm and only admits those regions large enough to be valid balls. It then processes these ball regions using circle-extraction and best-fit routines, leading to an accurate estimate of each ball's center location.

Once Deep Green has accurately identified the ball locations, it sends the circular subregions defining each ball to a color-indexing routine to determine the ball identities. It must know the exact identity (number) of each ball, as the formal rules for 8 Ball require nominating a ball and pocket for each shot. Offline, the system forms a 2D histogram in normalized RGB space for each of the 16 ball types from a collection of images of each ball, taken at different aspects and at various locations on the table. At runtime, it compares the color space histogram of each ball region with this database and uses a histogram similarity metric to classify the ball.

Despite strong similarities between the colors of different ball types, and reuse of colors among the stripes and

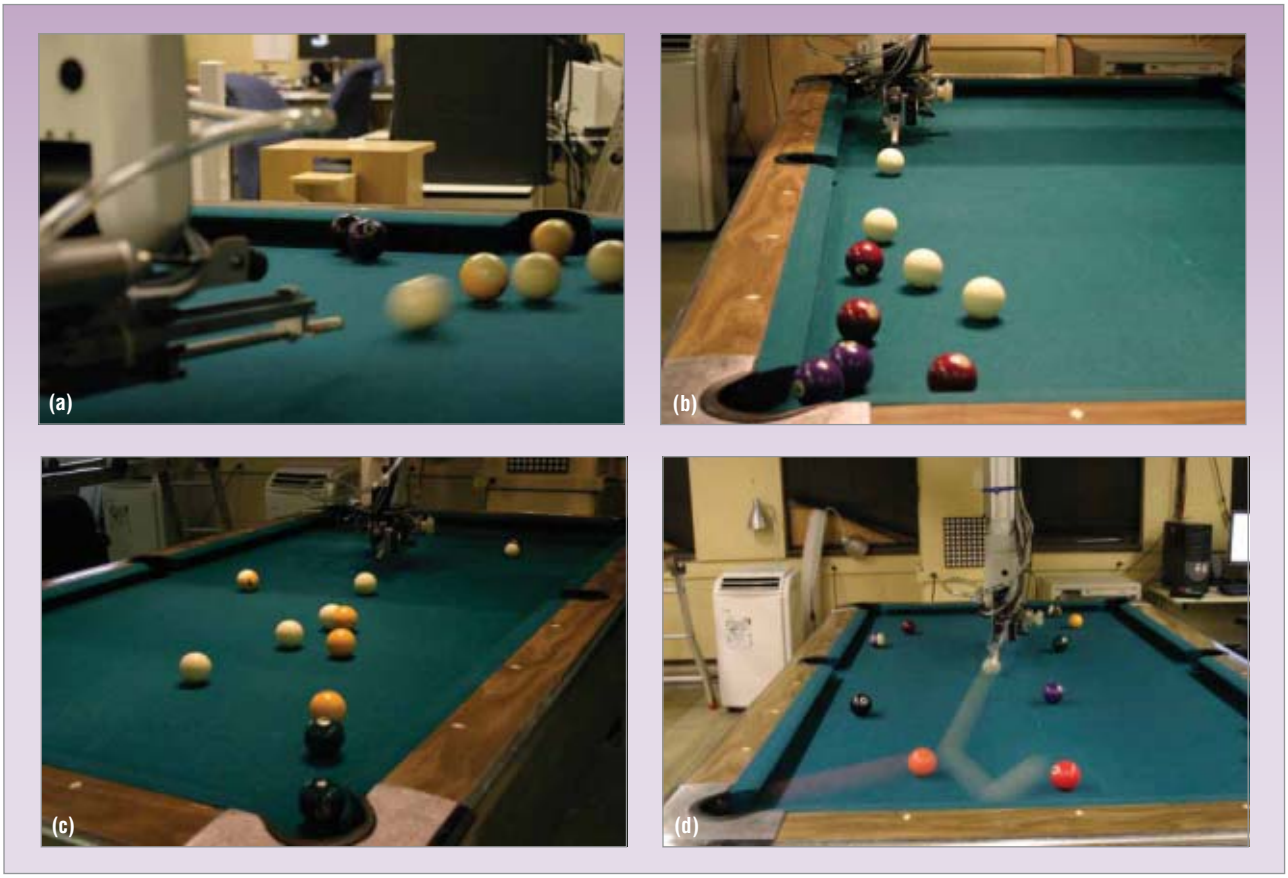


Figure 3. Example shots. (a) 9 ball in the side pocket—composite of three images. (b) Combination shot: 4 ball in the corner pocket, off of the 7 ball—composite of four images. (c) Combination shot: 6 ball in the corner pocket, off of the 1 ball—composite of four images. (d) 5 ball in the corner pocket—time-exposure image.

solids, the color-indexing method can reliably determine each ball's identity. Once the system has accurately localized and identified each ball, it can simulate the table state for shot planning.

Robot calibration

The challenge in using a standard gantry platform is its limited accuracy, as industrial robotics tend to be highly precise and repeatable but not terribly accurate. While it's possible to design a gantry robot with fine-grained accuracy, such a device would be expensive, delicate, and unlikely to maintain its accuracy while absorbing the impacts required to place shots. A more reasonable approach is to demand less accuracy from the primary positioning device and rely upon the vision system for calibration and correction.

One calibration technique involved both the LVS and GVS cameras.[9] We repeatedly positioned the robot over a series of circular patterns placed on the table surface. We then used the correspondence between the robot joint encoder values and the centers of the extracted circles within the GVS image to determine the functional relationship between the robot coordinate frame and the table plane. This technique reduced robot positioning error from the order of centimeters to within 0.6 mm on average, with a standard deviation of 0.3 mm.

Eye-in-hand visual servoing

While robot calibration rendered an improvement, a positioning accuracy of 0.6 mm is insufficient to successfully pot many long shots. It may be possible to further refine our calibration technique, successively unraveling the robot's many mysterious nonlinearities. However, the likely result of such an effort would be a very brittle system—any change in the system parameters, due to aging or other extrinsic conditions such as vibrations or temperature, would require a tedious recalibration.

To improve positioning accuracy, we have developed an eye-in-hand visual-servoing system in which the LVS camera is mounted on the end-effector with its optical axis pointing roughly along the direction of the cue. The LVS uses the known ball locations determined by the GVS as visual landmarks to detect and compensate for positioning errors accumulated during the gantry's coarse motion.

LVS correction. Consider the nearly perfect straight shot illustrated in Figure 4. In this GVS image, the inscribed line is defined by the extracted center locations of the

cue and object balls prior to placing the shot. The rendered circles are a sequence of three extracted positions of the object ball, at times t_0 to t_2, once the shot has been placed. The centers of these circles fall on or close to the line, indicating that the robot was positioned to make a very accurate straight shot. The final resting positions of the cue and object balls at time t_f also fall on this line, further supporting the shot's quality.

From the LVS's vantage, this is the *ideal line*. When the robot is servoed to its shot position, as determined by the GVS, it accumulates error. By analyzing the LVS image, and comparing the line connecting the current cue and object ball centers with the ideal line, the system can calculate transformations that correct for the robot positioning error.[10]

Figure 5a shows an LVS image acquired after the robot has been servoed to its shot position, using only the information from the GVS. The current (red) and ideal (green) lines aren't aligned, indicating positioning error. After the system executes the automatic alignment procedure, the current line overlaps almost exactly with the ideal line, as shown in Figure 5b, and the shot will therefore be very close to a perfect straight shot.

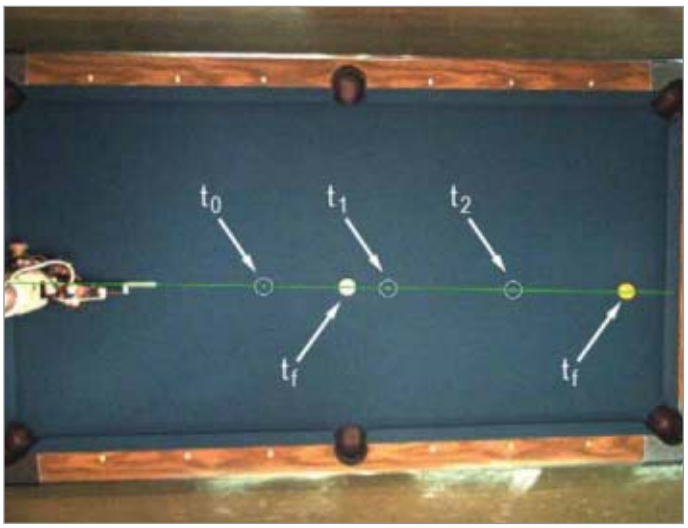


Figure 4. Straight shot. Intermediate object ball locations fall on a line defined by initial cue and object ball locations.

Alignment methods. We have developed two different methods to align the robot position with the LVS ideal line.[10] The simpler one is iterative and based entirely on 2D LVS image data. The other method uses knowledge of the 3D rigid transformation between the robot wrist coordinate reference frame and the LVS optical frame. This transformation, known as the *tool control frame* (TCF) matrix, is determined offline in a calibration stage.

Figure 6 plots the result of an experiment designed to characterize the performance of these two methods. A total of 90 straight shots were executed. Thirty of these shots used only information from the GVS and robot calibration, 30 more applied alignment using the image-based method, and the final 30 used the position-based method. We calculated the angular error of each shot by extracting the object-ball center locations at a number of (at least two) positions along their trajectories using the GVS and comparing the angle of this line with the line defined by the cue and object balls prior to placing the shot (similar to Figure 4).

We plotted the angular errors for each of the 3 sets of 30 shots in ascending order. Alignment using either method significantly reduced the angular error. Without alignment, the mean absolute error was 1.8 degrees. With alignment, the error was reduced by more than two thirds, to 0.51 degrees and 0.56 degrees for the image- and position-based methods, respectively. While the accuracy is similar for both alignment methods, the position-based method is approximately 40 percent faster. Once the straight shot is aligned accurately, the TCF matrix can be used to further rotate and translate the cue around the cue-ball center to execute a cut shot of any desired angle and spin.

GAMING

For those who play pool only casually, skill is the limiting factor, and sinking the current ball is usually the sole concern. For more advanced players, however, strategy

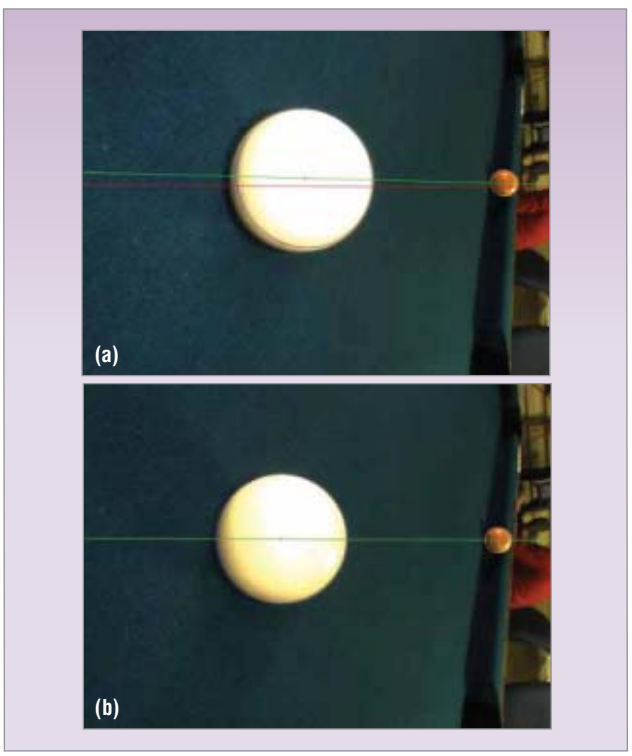


Figure 5. LVS correction. (a) Current (red) and ideal (green) lines before alignment. (b) After alignment, current and ideal lines overlap.

becomes a key element of the game, and professionals are known to plan five or more shots ahead for a given table state. For a robotic system to play competitively, it must therefore strategize computationally, which involves both predicting and planning future table states. This requires the interplay of physics simulation and search.

Physics simulation

To predict the table state after a shot so that subsequent shots can be planned, an accurate physics model is necessary. Spin is an essential element of the game, and imparting spin on the cue ball by displacing and angling the cue at impact is a technique used to control the interaction and placement of balls following a shot.[11] The physics model therefore involves conserving not only linear but also angular momentum.

We have developed a physics simulator that predicts a shot's outcome from a derived physics model.[12] Unlike physics simulators that use the more common numerical integration approach, our method operates in the continuous domain, predicting the times of pending events such as collisions or transitions between motion states. Our technique returns an exact analytic solution based on a parameterization of the separation of two moving balls as a function of time. The resulting equation is a quartic polynomial that can be solved either iteratively or in closed form to determine the collision time. A similar derivation exists for other events, such as ball-rail and -pocket collisions and transitions from sliding-to-rolling and rolling-to-stationary states.

Compared to integration, our approach is more

- accurate, requiring no discrete time step; and
- time efficient, requiring approximately two to three orders of magnitude fewer computations per shot.

This added efficiency is especially important when the physics simulator is used in expanding a game tree, as many different shots—sometimes tens of thousands or more—might need to be simulated prior to making a decision.

Our physics simulator was the basis for the Computational 8 Ball Tournaments at the 10th and 11th International Computer Olympiads.[13] These tournaments let teams develop different strategy engines and compete using the common physics simulator.

One consideration in modeling the physics was shot noise. When a human or robotic player takes a shot, error in the cue's position and velocity makes each shot nonideal. To make the simulation more realistic and the competition more challenging, we added zero-mean random Gaussian noise to each of the five shot parameters that determine the outcome of a shot: two angles (θ,ϕ), two offsets (a, b), and the striking speed V.[14] The sigma values of each distribution were empirically determined to cause one missed shot every 10 shots on average, a success rate similar to that of advanced human play. When planning a shot for robotic play, a noise model based on the robot's calibrated positioning accuracy can be used to determine the probability of a given shot's success.

Search

With the physics simulator's ability to predict a shot's outcome, it's then necessary to evaluate many possible shot sequences to determine the best shot to place given the current table state. Our approach to this search is based on the minimax game tree used in games like chess and checkers.[15,16] While the basic concept is the same as in chess, one difference is that pool is played in a continuous, rather than a discrete, domain. The size of the search space for any particular shot is therefore truly infinite, rather than the huge but finite search space of chess.

Another unique consideration in pool is shot noise. In practice, each of the five shot parameters has an element of uncertainty that can be modeled as a probability distribution. For this reason, we have adapted the expectimax search tree, which has been applied to games like backgammon that have a probabilistic component. Because pool is played in a continuous domain, the chosen tree search algorithm incorpo-

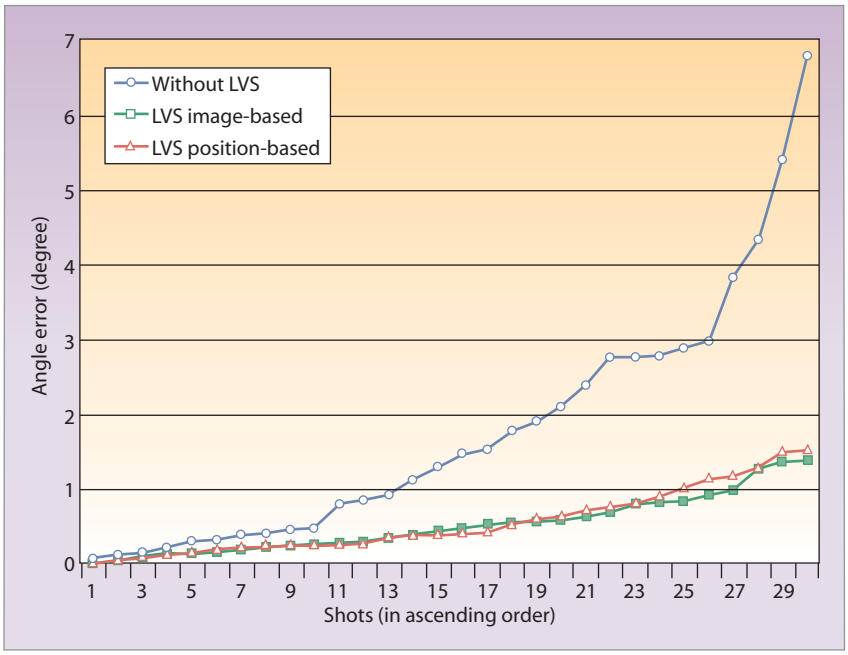


Figure 6. Angular error in straight shot tests with LVS correction. Alignment reduced the error by more than two thirds, to 0.51 degrees and 0.56 degrees for the image- and position-based methods, respectively.

Table 1. Summary across search depths for zero-, low-, and high-noise tournaments.

Noise	Player	Average wins (percent)	Average points scored	Average point differential	Average misses (percent)	Average ball-in-hand (percent)
Zero	Greedy	9.9	771.6	–1,093.3	0.0	10.3
	All depth 1	61.1	1,390.8	278.4	0.0	2.5
	All depth 2	79.9	1,622.5	814.9	0.0	2.5
Low	Greedy	19.9	963.1	–791.0	6.3	12.0
	All depth 1	62.7	1,458.8	323.9	2.6	3.6
	All depth 2	67.4	1,523.9	467.1	1.6	3.4
High	Greedy	36.5	1,301.7	–314.6	11.8	14.3
	All depth 1	54.8	1,484.4	114.2	8.9	10.4
	All depth 2	58.7	1,519.9	200.4	9.3	9.0

rates statistical sampling to account for uncertainty in shot execution. The utility of a future table state is weighted by its probability of occurrence, and the weighted utilities of the children of each node are combined when considering which path to traverse.

Empirical evaluation of strategic play

To explore the benefits of strategic play in pool, we executed a set of experiments using this tree-search framework.

Methodology. We simulated a series of 8 Ball tournaments involving 19 competitors, all with identical shot-generation algorithms. Eighteen of the competitors used different tree-search depths, tree-scoring variations, and evaluation-function variations; the 19th used a depth-zero "greedy" shot-selection algorithm based solely on the probability of the current shot's success with no regard for the resulting table state or future shots. This greedy player had the same skill level as the other competitors but thought like an amateur.

Three tournaments were played with three different noise models reflecting the players' technical skill level. For the high-noise model, about 80 percent of balls were sunk as planned; for low noise, about 90 percent; and for zero noise, all shots were executed exactly as planned. All players in each tournament used the same noise model and search algorithm. Each tournament therefore isolated performance as a function of tree-search depth and evaluation-function variation. A search depth of 1, for example, considers not only the current shot but also all shots resulting from the current shot. The various scoring and evaluation functions differed in how they rated a leaf node's utility as well as in how they combined the information from child nodes in propagating back up the tree.

This is similar to comparing two human players by categorizing their play in two areas:

- technical skill—precision in executing shots; and
- level of strategic play—how far ahead in the game the player looks, and how the player controls the cue ball position for the next shot.

We examined numerous combinations of tree-scoring variations—Monte Carlo, probabilistic, or success-weighted—and evaluation-function variations: average, maximum, or weighted. Within each tournament, the players with common search algorithm/evaluation functions (but varying search depth) played 200-game matches against one another and against the greedy player in a round-robin format. The winning player of each game received a total of 10 points, and the losing player received one point for each pocketed ball of its color group (stripes or solids), for a maximum of seven points. The match score was the sum of the game scores.

Results. Table 1 summarizes the results from these experiments. Players are ranked by their overall performance by averaging the percentage of games won, points scored, point differential, miss rate, and percentage of shots resulting in a ball-in-hand. The percentage of shots resulting in a BIH indicates not only how often a player fouled, but more importantly how often it left itself with no shot. The greedy player was more heavily penalized by this setting because it never considered the table state resulting from its chosen shot.

In the zero-noise tournament, the deeper-searching players consistently outplayed their shallower-searching competitors. For a given search type/evaluation function variant, the depth 2 player always defeated the greedy player easily and then defeated the depth 1 player in turn. The greedy player was defeated in all matches in the zero-noise tournament, winning at best 16.5 percent of the games in its match against one player. Against the greedy player, all of the depth 2 players scored more wins with a higher point differential than the corresponding depth 1 player.

Look-ahead. Positional play in the form of look-ahead is clearly an important consideration in pool. Choosing the easiest shot, or the shot with the highest probability

of success, doesn't result in a competitive player; planning strategically using look-ahead does. These results mirror the expectation for human players similarly characterized by technical skill and level of strategic reasoning. Players are always limited by their technical skill, regardless of how strategically they plan shots. However, for sufficiently skilled players, the benefits of strategic reasoning and cue-ball placement in the form of look-ahead always dominate over less strategic play.

While these experiments have evaluated look-ahead only to a depth of 2, the benefits of look-ahead should continue to be apparent for search depths up to 8, at which point all game tree branches will have terminated, with all balls sunk and the game completed. In practice, expanding the game tree to greater depths can be quite time expensive, and so tournament competitors have restricted their searches to depths of 2 or 3.

ADVANTAGES OF MACHINE PLAY

In many ways, pool is an ideal game for automation. A great deal of human pool instruction and practice is oriented toward establishing an accurate and repeatable stroke. Machines routinely outperform humans at positioning accuracy and repeatability, and they function consistently, without the performance-degrading effects of muscle fatigue. They also aren't susceptible to psychological pressure, a significant source of variation and failure in human play.

In addition, a machine like Deep Green can sense the balls' absolute metric locations in the table coordinate reference frame. Humans can ascertain the balls' geometric arrangement based on their relative positions on the table, allowing them to plan and execute challenging shots, but in certain situations even skilled humans have difficulty perceiving the correct angles. For example, shots that involve multiple banks are inherently difficult to perceive, and humans often use inexact systems based on table landmarks (diamonds) to augment their perception. In contrast, the machine resolves the metric location of all balls and table elements such as rails and pockets. This allows for more exact geometric planning, and enhances the machine's ability to predict a shot's outcome.

Another advantage of the machine is its computational simulation of the table's physics. Most human players rely on an intuitive understanding of this aspect of the game. Typically with little or no formal knowledge of physics, they develop heuristics to predict the subsequent table state that results from the multiple interactions of any particular shot. While often useful, these heuristics have limited fidelity. In contrast, the machine has an executable physics model and, so long as a handful of parameters have been estimated through calibration, can use a physics simulator to predict the resulting table state both accurately and efficiently.

Moreover, the cue end-effector provides precise control of stroke speed. The electromagnetic linear actuator responsible for the forward motion of Deep Green's stroke has a dedicated digital control unit that can be commanded in either position or velocity modes. The cue's speed can range from almost stationary to approximately 3 m/s, with an average error of approximately 0.1 percent. In contrast, humans tend to strike with one of six speeds: slow, medium-slow, medium, medium-fast, fast, or break. The added graduation in controlling cue speed translates to an increased ability to place the cue ball and predict and control the table state.

Once the mechanics of placing a shot have been mastered, pool becomes a strategic game, and here too the machine has a potential advantage. The essence of pool strategy is the ability to look ahead and predict the table's state following a potential shot or series of potential shots. This same capability lets computers outperform people at chess and other games recently believed to be only within the realm of human mastery.

NEED FOR INTELLIGENCE

The Deep Green project has inspired polar opposite responses on the degree of difficulty required to attain our goal. Some people who are familiar with technology but not with pool have regarded it as a straightforward task, requiring only standard robotic techniques to provide a solution. In contrast, proficient players who have no special relationship with technology tend to argue that pool is a distinctly human activity, requiring human intelligence and skill, and that automating it is impossible.

Our view lies somewhere between these two extremes. We believe that developing a robotic system to play pool competitively against a proficient human opponent is achievable. The technical problems are both interesting and sufficiently challenging to motivate advanced research, but not so difficult as to evade a meaningful solution.

Another question that Deep Green raises is whether computational intelligence is necessary for robotic pool. Isn't an accurate positioning system and simple shot planning based purely on geometry sufficient? There are two answers to this question. First, accurate positioning of a standard gantry robot is itself a challenging goal requiring sensor-based methods for calibration and correction. Second, even if perfectly accurate positioning were possible, it's still advantageous to play strategically and plan ahead a number of shots, as evidenced by our experiments with zero-noise tournaments.

Deep Green currently plays at a better-than-amateur level, planning and executing difficult combination and rail shots from across the table. It has pocketed runs of four consecutive balls, and it's only a matter of time before it can consistently run the table.

Several research challenges must be addressed to advance the system further. The most difficult will emerge in competing against proficient human opponents. Humans are

crafty competitors, able to efficiently recognize and exploit weaknesses in their opponents. To play at a competitive level, Deep Green must incorporate insights from machine-learning and opponent-modeling techniques.

Acknowledgments

The authors thank Precarn Inc., the Institute for Robotics and Intelligent Systems, the Canada Foundation for Innovation, and the Natural Sciences and Engineering Research Council of Canada for their financial support, and Elisha Hardwick for her photographic work. They also thank the many students who have contributed long hours to the development of the Deep Green system.

References

1. Sporting Goods Manufacturing Assoc., *State of the Industry Report,* 2005.
2. S.W.S. Chang, "Automating Skills Using a Robot Snooker Player," doctoral dissertation, Univ. of Bristol, Bristol, UK, 1994.
3. R. Popper, "Bring Back the Snooker-Playing Robot!," *The Guardian*, 7 Aug. 2007, p. 3.
4. M.E. Alian et al., "Roboshark: A Gantry Pool Player Robot," *Proc. 35th Int'l Symp. Robotics* (ISR 04), 2004; www.cs.sfu.ca/~psabzmey/personal/publ/papers/alian_roboshark_isr04.pdf.
5. S.C. Chua et al., "Performance Evaluation of Fuzzy-Based Decision System for Pool," *Applied Soft Computing,* Jan. 2007, pp. 411-424.
6. Z.M. Lin, J.S. Yang, and C.Y. Yang, "Grey Decision-Making for a Billiard Robot," *Proc. 2004 IEEE Int'l Conf. Systems, Man and Cybernetics,* vol. 6, IEEE Press, 2004, pp. 5350-5355.
7. L.B. Larsen, M.D. Jensen, and W.K. Vodzi, "Multimodal User Interaction in an Automatic Pool Trainer," *Proc. 4th IEEE Int'l Conf. Multimodal Interfaces* (ICMI 2002), IEEE CS Press, 2002, pp. 361-366.
8. H. Denman, N. Rea, and A. Kokaram, "Content-Based Analysis for Video from Snooker Broadcasts," *Computer Vision and Image Understanding,* Nov./Dec. 2003, pp. 176-195.
9. F. Long et al., "Robotic Pool: An Experiment in Automatic Potting," *Proc. 2004 IEEE/RSJ Int'l Conf. Intelligent Robots and Systems* (IROS 2004), vol. 3, IEEE Press, 2004, pp. 361-366.
10. J. Lam, "Eye-in-Hand Visual Servoing to Improve Accuracy in Pool Robotics," master's thesis, Dept. Electrical and Computer Eng., Queen's Univ., Kingston, Canada, 2007.
11. D.G. Alciatore, *The Illustrated Principles of Pool and Billiards,* Sterling, 2004.
12. W. Leckie and M. Greenspan, "Pool Physics Simulation by Event Prediction 2: Collisions," *Int'l Computer Games Assoc. J.,* Mar. 2006, pp. 24-31.
13. M. Greenspan, "Pickpocket Wins Pool Tournament," *Int'l Computer Games Assoc. J.,* Sept. 2006, pp. 153-156.
14. W. Leckie and M. Greenspan, "Pool Physics Simulation by Event Prediction 1: Motion Transitions," *Int'l Computer Games Assoc. J.,* Dec. 2005, pp. 214-222.
15. W. Leckie and M. Greenspan, "Monte-Carlo Methods in Pool Strategy Game Trees," *Proc. 5th Int'l Conf. Computers and Games* (CG 06), LNCS 4630, Springer, 2007.
16. M. Smith, "PickPocket: A Computer Billiards Shark," *Artificial Intelligence*, Nov. 2007, pp. 1069-1091.

Michael Greenspan *is an associate professor in the Department of Electrical and Computer Engineering and in the School of Computing, Queen's University, Kingston, Ontario, Canada, and is spending the year as a visiting scientist at the University of Coimbra, Portugal. His research focuses on problems in computer vision and computer gaming. Greenspan received a PhD in systems and computer engineering from Carleton University, Ottawa. Contact him at michael.greenspan@queensu.ca.*

Joseph Lam *is a PhD candidate in the Department of Electrical and Computer Engineering at Queen's University. His research interests include computer vision and visual servoing. Contact him at jctlam@gmail.com.*

Marc Godard *is a bachelor's student in the School of Computing, Queen's University. His research interests include computer vision, cognitive models, and prediction algorithms. Contact him at 4mg12@qlink.queensu.ca.*

Imran Zaidi *is a bachelor's student in the School of Computing, Queen's University. Contact him at 3aiz@qlink.queensu.ca.*

Sam Jordan *is a master's student in the Department of Electrical and Computer Engineering at Queen's University. His research interests include robotics, augmented reality, and human-computer interaction. Contact him at 3sj1@queensu.ca.*

Will Leckie *is an electro-optics hardware designer with Nortel at its Carling Campus in Ottawa. His research interests include robotics and artificial intelligence. Leckie received a master's degree in electrical and computer engineering from Queen's University. Contact him at will.leckie@ece.queensu.ca.*

Ken Anderson *is a software engineer at Larus Technologies in Ottawa. His research interests include robotics, single-agent search, and computer games. Anderson received a master's degree in computing science from the University of Alberta. Contact him at anderson@cs.ualberta.ca.*

Donna Dupuis *is a master's student in the Department of Electrical Engineering at the University of British Columbia, Vancouver, British Columbia, Canada. Her research interests include computer vision, robotics, and artificial intelligence. Contact her at dupuisd@mech.ubc.ca.*

Harnessing Digital Evolution

Philip McKinley, Betty H.C. Cheng, Charles Ofria, David Knoester, Benjamin Beckmann, and Heather Goldsby
Michigan State University

In digital evolution, self-replicating computer programs—digital organisms—experience mutations and selective pressures, potentially producing computational systems that, like natural organisms, adapt to their environment and protect themselves from threats. Such organisms can help guide the design of computer software.

Nearly 150 years ago, Charles Darwin explained how evolution and natural selection transformed the earliest life forms into the rich panoply of life seen today. Scientists estimate this process has been at work on Earth for at least 3.5 billion years.

But we remain at the dawn of evolution in another world: the world of computing. There, evolution helps humans solve complex problems in engineering and provides insight into the evolutionary process in nature. As computing power continues to increase, researchers and developers apply evolutionary algorithms to an ever-widening variety of problems. As the "Evolution in a Computer" sidebar shows, evolutionary computation methods such as genetic algorithms have already achieved considerable success, rivaling and surpassing human designers in problem domains as wide-ranging as flash memory sticks and aircraft wings.

We are investigating how to harness the power of evolution to help construct better computer software. The increasing interaction between computing technology and the physical world motivates this work. Systems must adapt to their environment, compensate for failures, optimize performance, and protect themselves from attacks—all with minimal human intervention.[1,2]

To design robust and resilient computational systems, we can take inspiration from nature. Living organisms have an amazing ability to adapt to changing environments, both in the short term through phenotypic plasticity and in the longer term through Darwinian evolution. Indeed, no existing cybersystem rivals the complexity of Earth's biosphere, yet life on Earth has evolved to not only deal with this complexity but to thrive on it.

Many researchers have studied how to use the characteristics of natural systems to design better computing systems. One approach mimics the behaviors of social insects and other species. However, while such biomimetic methods have shown promise in controlling fleets of unmanned robotic systems and in other applications, they can only codify behaviors observed in nature *today*. Purely biomimetic approaches seek to imitate the results of evolution, but they do not account for the process of natural selection that produced those behaviors.

For example, we can design the control software on a microrobot so that it mimics certain behaviors found in ants. However, while the robot might possess some physical characteristics reminiscent of an ant, the differences vastly outnumber the similarities. On the other hand, if we had the ability to *evolve* the control software, taking into account the capabilities of the robot and the characteristics of its environment, new behaviors might emerge that more effectively control the robot.[3]

DIGITAL PETRI DISH

Digital evolution gives us this power, and we are investigating how it can aid us in designing robust computational systems. Digital evolution is a form of evolutionary computation in which self-replicating computer

Published by the IEEE Computer Society

Evolution in a Computer

Evolutionary computation,[1] a subfield of computer science, applies the basic principles of genetic evolution to problem solving. EC is based on evolutionary biology and extends into many other fields, including artificial life. In general, an EC system contains one or more populations of individuals that compete for resources in a computational environment. These individuals produce offspring according to their fitness, which often depends on the problem domain. The most well-known evolutionary computation method is the genetic algorithm.[2] In this iterative search technique the individuals in the population are encodings of candidate solutions to an optimization problem. In each generation, the fitness of every individual is calculated, and a subset of individuals is selected, recombined, or mutated, and moved to the next generation.

Genetic programming[3] provides a related method in which the individuals are actual computer programs. These approaches and other EC methods have been used to solve complex problems, in some cases producing patentable designs.[3] The annual Genetic and Evolutionary Computation Conference gives awards for human-competitive results produced by genetic and evolutionary computation (www.geneticprogramming.org/hc2007/cfe2007.html).

While the broad field of evolutionary computation has been studied extensively since the 1960s, the subfield of digital evolution is much younger. Self-replicating digital organisms can be traced to the game *Core War*, which Steen Rasmussen extended in 1990 into a system he called *Core World*. Soon after, Thomas Ray designed *Tierra*, which used a streamlined and fault-tolerant genetic language.

In 1993, Charles Ofria, Chris Adami, and C. Titus Brown began developing the Avida digital-evolution platform[4] at the California Institute of Technology. In Avida, each program lives in its own address space, unlike *Tierra*'s shared address space. This enhancement increased the power of digital evolution as an experimental tool. Avida has since been used to conduct pioneering research in the evolution of biocomplexity, with an emphasis on understanding the evolutionary design process in nature. In addition to providing a tool for biologists, digital evolution provides an open-ended search technique for problems in science and engineering.

References

1. K.A. De Jong, *Evolutionary Computation: A Unified Approach*, MIT Press, 2002.
2. J.H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence,* Univ. Michigan Press, 1975.
3. J.R. Koza et al., *Genetic Programming IV: Routine Human-Competitive Machine Intelligence, Genetic Programming*, Springer, 2005.
4. C. Ofria and C.O. Wilke, "Avida: A Software Platform for Research in Computational Evolutionary Biology," *J. Artificial Life*, vol. 10, 2004, pp. 191-229.

programs evolve within a user-defined computational environment.[4] These *digital organisms* receive limited resources whose use must be carefully balanced if they are to survive. As organisms replicate, instruction-level mutations produce variation within the population. Over generations, natural selection can produce instruction sequences that can realize complex behaviors, sometimes revealing unexpected and strikingly clever strategies for solving problems.

Our work uses and extends the Avida digital evolution platform. Figure 1 depicts an Avida population and the structure of an individual organism. Each digital organism consists of a circular list of instructions—its genome—and a virtual CPU, which executes the instructions. An Avida environment comprises several cells, each of which can contain at most one organism, or *Avidian*. When an Avidian replicates, the system places the offspring in a randomly selected cell, terminating any previous inhabitant. Organisms can send messages to each other, produce and consume resources, and sense and change their environment's properties. Through these interactions, an organism can gain or lose virtual CPU cycles, affecting how fast it executes instructions.

The virtual CPU architecture used in most of our studies is simple, containing three general-purpose registers {AX, BX, CX}, two general-purpose stacks {GS, LS}, and four special-purpose heads. These heads serve as pointers into the organism's genome and resemble a traditional program counter or stack pointer. The instruction set for this virtual CPU is Turing-complete, and therefore, theoretically, it can realize any computable function. Available instructions perform basic computational tasks (addition, multiplication, and bit-shifts), control execution flow, enable communication, and allow for replication. Although the instruction set resembles a traditional assembly language, it is designed so that random mutations (inserting, deleting, or changing instructions) will always yield a syntactically correct program.

Avidians receive virtual CPU-cycle rewards for performing user-defined tasks, generally defined in terms of the organisms' externally visible behaviors—their phenotype. For example, a task might require the organism to

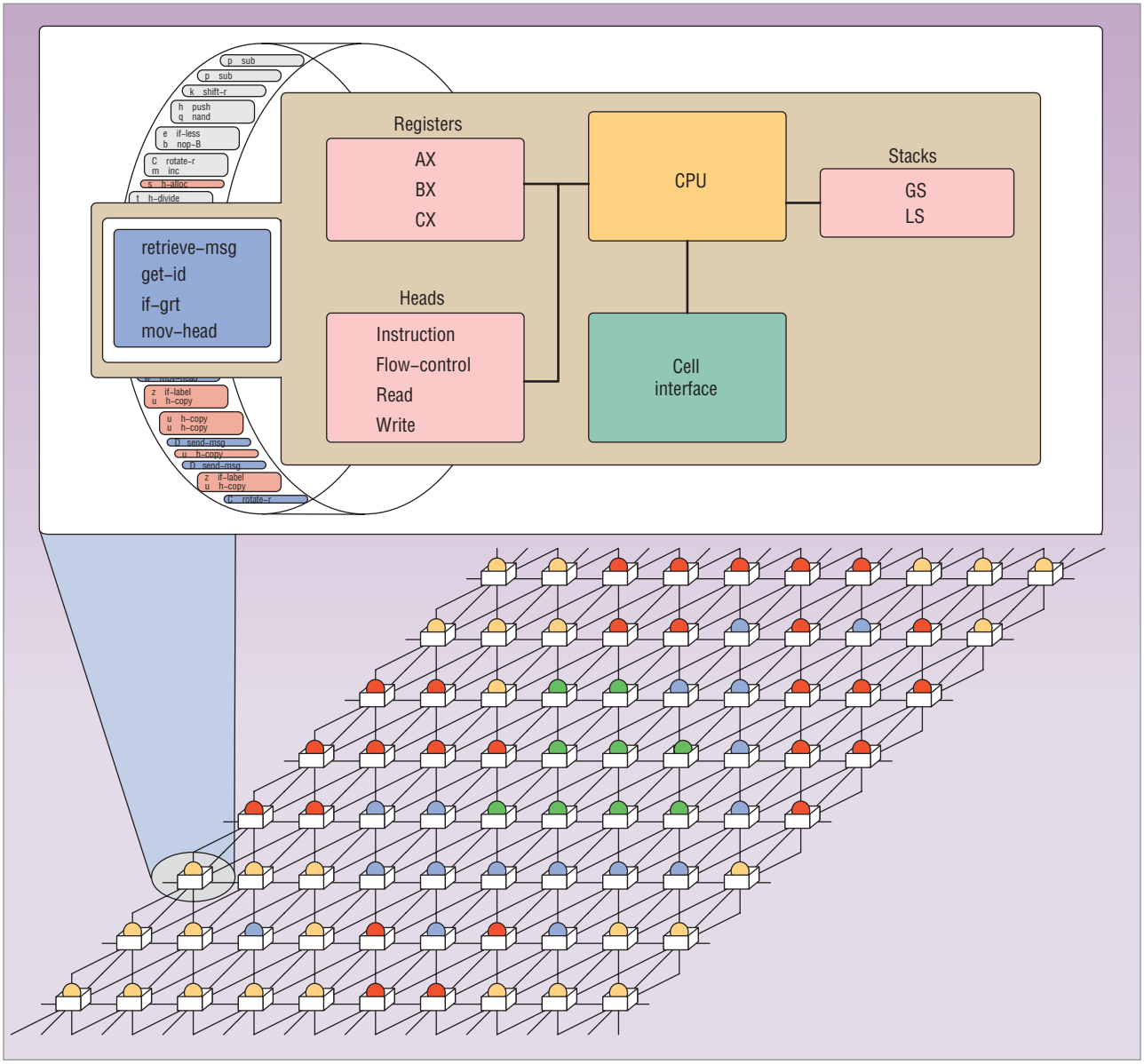


Figure 1. Elements of the Avida digital evolution platform: (top) structure of an individual organism and (bottom) a population of digital organisms in an 8 × 10 grid of cells. Different colors represent organisms with different genomes.

perform a particular mathematical or logical operation and output the result, or communicate with neighboring organisms in a manner helpful to distributed problem solving. Tasks create selective pressures in the population, favoring genomes where mutations have produced sequences of instructions that complete tasks.

In our studies, evolved code segments for completing tasks vary in length from a few instructions to several tens of instructions. An evolved solution might not be optimal when considering the task in isolation, but will likely have other properties that make it well-suited to its environment—robustness to mutation, for example. Moreover, code for completing tasks cannot mutate into the genome at the expense of replication, which is the only way organisms can pass their genetic material to future generations. Avidians that are most successful—those that replicate faster, or perform user-defined tasks—are more likely to spread throughout and eventually dominate the population. Indeed, Avida satisfies the three conditions necessary for evolution to occur[5]: replication, variation (mutation), and differential fitness (competition). Avida does not simulate evolution—it is an *instance* of evolution.

Researchers created the Avida platform primarily for studying evolution in nature. Observing evolution in digital organisms lets users address questions difficult or impossible to study with organic life forms, such as explicitly disabling selected mutations and observing the effects, or analyzing the genomes of all organisms along a particular evolutionary path. For example, Richard

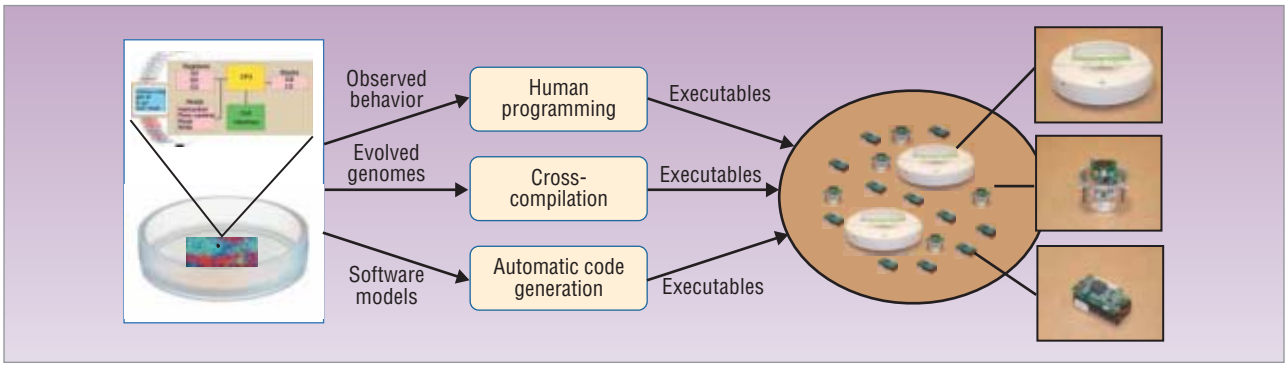


Figure 2. Different ways the authors use Avida to help develop computer software for robots and sensors. Three target platforms appear at the right of the figure and are, from top to bottom, the iRobot Create robot, e-puck educational robot, and MICA mote and sensor board.

Lenski and colleagues[6] used Avida to demonstrate that evolution can produce complex features by combining previously evolved "building blocks," helping to answer a long-standing scientific question posed by Darwin. However, Avida is an extensible platform that lets researchers and developers apply digital evolution to many different problem domains, including nonbiological ones. The user can completely customize many Avida features—including the virtual CPU architecture, instruction sets, and tasks. Today, a user with a modest compute cluster can explore hundreds of populations, totaling millions of generations, in a single day. Effectively, Avida provides the user with a digital "petri dish" for creating and analyzing new computational behaviors.

APPLICATION TO SOFTWARE DESIGN

Among other applications in science and engineering, digital evolution enables a fundamentally new approach to software design, whereby developers can actively explore new program behaviors and prospective pathways for complex software systems, all during the initial design. As depicted in Figure 2, we can apply Avida in at least three different ways to create software.

First, similar to biomimetics, behaviors that evolve in silico can provide insight into the design of new algorithms and protocols. Moreover, since digital organisms live in an environment that can be user-configured, exploration of behaviors is not limited to those found in the natural world. Novel strategies revealed through digital evolution can be codified in a traditional programming language and deployed in hardware. For example, digital evolution might yield energy-conserving behaviors that can be programmed and deployed in sensor networks.

Second, since the genomes of digital organisms are programs, they can be cross-compiled and executed directly atop hardware. For example, our group has recently developed a tool that converts Avida genomes to C code, which can be compiled and executed on a variety of devices, including sensor nodes and mobile robots. Such technologies let us test Avida-generated behaviors—such as cooperative communication operations and group-oriented mobility control—in the real world.

In the third approach, instead of evolving the software itself, we can evolve organisms that generate software artifacts. For example, we have applied digital evolution to the problem of generating and extending software design models to satisfy requirements. Specifically, Avidians act as generators and evolve to construct in-memory representations of state diagrams describing the system's behavior. Organisms can gain virtual CPU cycles by constructing state diagrams that meet requirements specified by the user, including scenarios that should be supported and properties that should be satisfied. If successful, natural selection produces a population of organisms that generate increasingly better solutions. Existing software engineering tools can be used to translate the resulting models into code.

ONGOING STUDIES

Our initial investigations focus on evolving behaviors needed in computing systems that interact with the physical world: cooperative communication, energy conservation, and adding new functionality to an existing system. When exploring a particular problem, we typically execute several batches of Avida runs, each with a different mix of tasks, then analyze the evolutionary process and resulting behaviors. A batch typically contains 20 runs, each of which starts with the same default organism capable only of self-replication. All other behaviors must enter the genome through mutations. Since each run within a batch starts with a different random number seed, the populations take different evolutionary paths.

Cooperative communication

The first study addresses the evolution of cooperative communication algorithms. Sensor networks often employ complex distributed operations such as multicasting, gathering sensed data, and detecting and

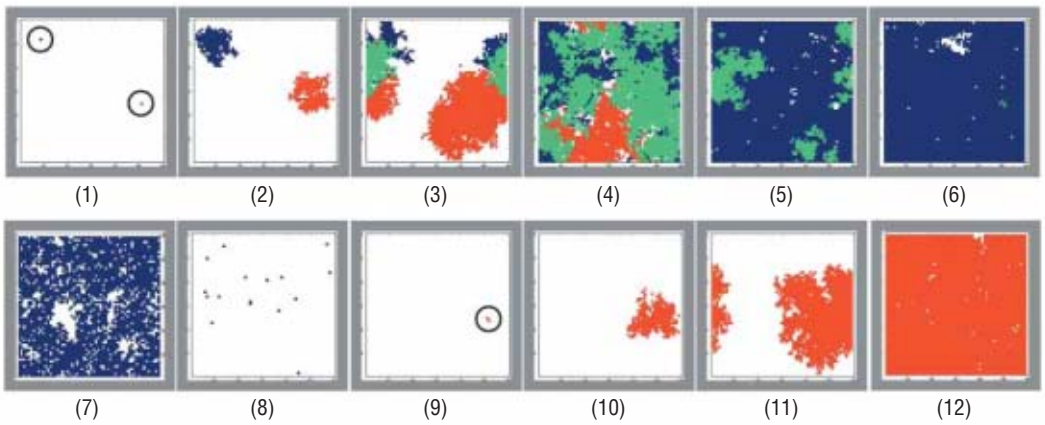


Figure 3. Snapshots of an Avida population in a 60×60 grid. The snapshots demonstrate distribution of the largest sensed value (blue) and, when that value resets, the next-largest value (red). Organisms sending both messages appear in green.

responding to events of interest. Unfortunately, many traditional algorithms for solving these problems are brittle when deployed in dynamic environments, and improvements are limited to the methods considered by human designers.

On the other hand, digital evolution provides a means to explore a larger solution space, potentially discovering algorithms more likely to remain effective even under extremely adverse conditions. For digital organisms to thrive in highly dynamic environments such as Avida, where organisms continually replace one another, they must evolve resilient solutions.

Consider the evolution of a particular distributed problem-solving task. We assign each cell in the environment a random 32-bit identifier, which a resident organism can sense using the GET-ID instruction. The organisms must determine the largest sensed value and distribute it throughout the population. Performing this operation could provide a basis for a leader-election algorithm, or a wireless sensor network could use it to obtain and distribute the maximum sensed value.

We designed a set of tasks to reward Avidians with more virtual CPU time for exhibiting cooperative behaviors and to penalize them with less when they did not. Over time, the population evolved to identify the largest value. To further assess the robustness of their solution to the problem, once the largest value had been discovered, we removed it from the population. This forced the population to continually search for the maximum value.

Figure 3 shows snapshots of a population that evolved these behaviors. The snapshots show the spread of two different values. Each snapshot identifies which organisms send the largest cell ID (blue), which send the second-largest cell ID (red), and which send both (green). By frame 6, nearly all organisms are sending messages that carry the largest cell ID; a few organisms near the cell with the second-largest ID send both. We reset the largest cell ID just prior to frame 7. As shown, the transmission of that cell ID dies out quickly. The population, however, recovers and proliferates messages that carry the new largest cell ID.

Figure 4 shows the dominant genome from the population exhibiting this behavior. This particular genome comprises 85 instructions, of which 11 are responsible for the desired behavior, 22 implement the organism's replication cycle, one instruction is shared, and 51 instructions—or 60 percent of the genome—are neutral mutations that do not affect the organism's phenotype. Genomes of living organisms, including humans, also contain large percentages of "junk DNA," the role of which researchers do not completely understand, but which might include serving as building blocks for new functionality.

Interestingly, this particular genome has a spin-wait near the top of the highlighted code segment, which effectively makes the organism's replication dependent upon receiving a message that carries a cell ID larger than its own. Organisms with this genome have evolved to the point where they depend upon other organisms' behavior for their survival: If an organism does not receive a message that has a data field larger than its own cell ID, it will not reproduce.

Energy management

Mobile devices with limited battery resources must conserve energy. For example, communication traffic flowing through an ad hoc wireless network directly affects the energy consumption at individual nodes, and excessive or disproportionate energy consumption can lead to node failure and possibly network partitioning. Determining the optimal energy management strategy in such situations involves many factors—such as dynamic flows, physical topology, movement constraints, security concerns, and energy consumption—and a multitude of possible scenarios. This part of our research investigates whether digital evolution can yield energy-efficient algo-

rithms and protocols that perform well under dynamic and adverse conditions.

Our early studies focus on the evolution of sleep behavior in digital organisms.[7] We subjected populations of Avidians to an environment with a slowly diminishing resource and recorded their ability to adapt to the changing environment using sleep instructions. These instructions let organisms enter a low-energy state that lasts for multiple CPU cycles. Avidians were rewarded for performing simple computational tasks—logic operations—but only when this resource, the digital equivalent of sunlight, was available. The resource was available a percentage of each 256-time-step Avidian day, but that percentage declined with each passing year of 500 Avidian days.

We observed that Avidians adapted to use sleep instructions effectively, despite the risk that a sleeping organism might be replaced before it reproduced. Examination of genomes showed that some populations evolved a behavior we anticipated, where organisms would sleep for short intervals and periodically wake to check for the resource. However, a majority of the populations evolved an unexpected behavior, the equivalent of a biological alarm clock, that adjusted the length of the gestation cycle to synchronize with the resource's availability. Moreover, experiments revealed that organisms evolved to start sleeping just before the resource went away and—just prior to the return of the resource—to awaken and begin preparing data to be used in tasks.

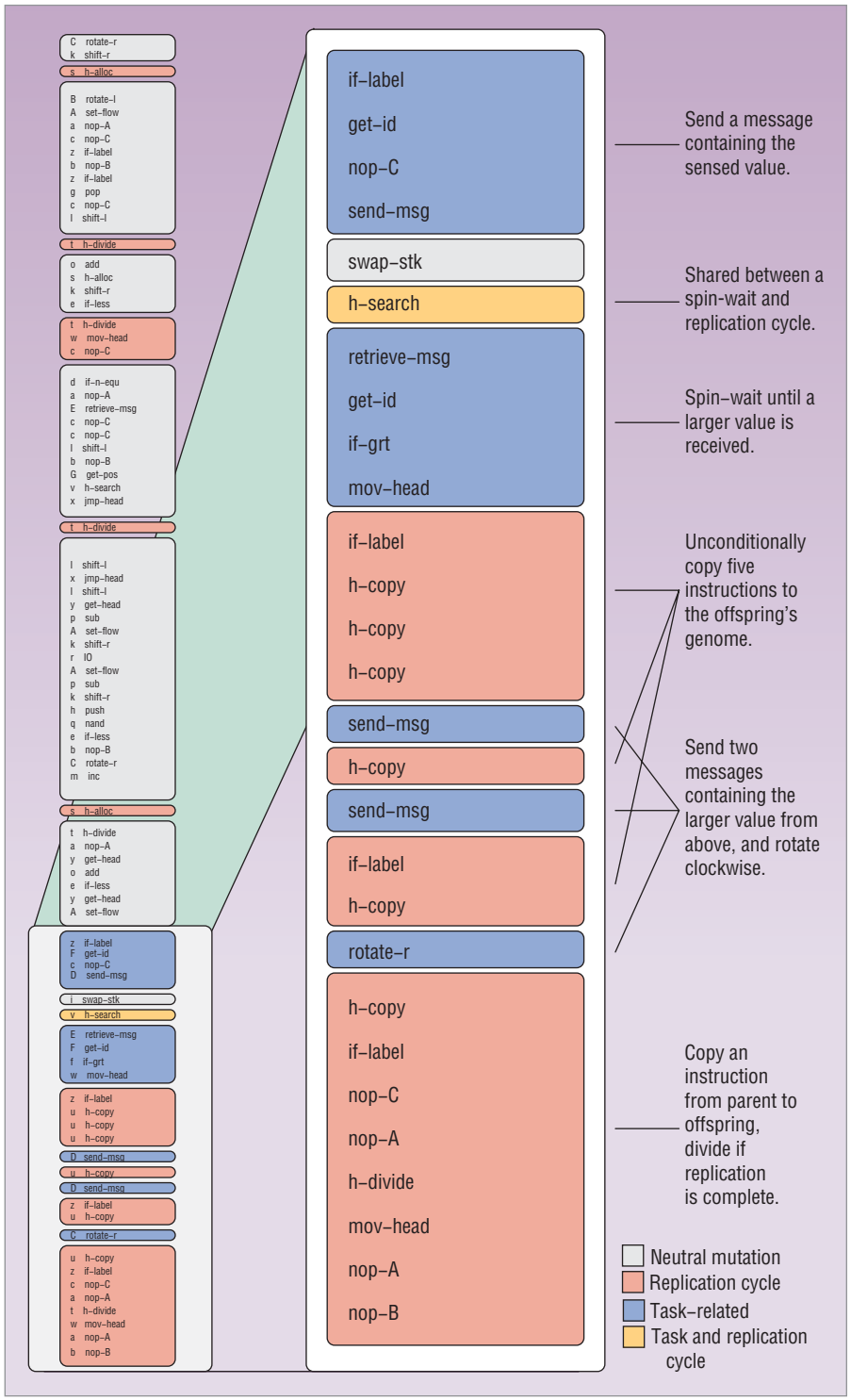


Figure 4. Dominant genome in a population that evolved to identify and distribute the largest cell ID. The full genome appears on the left. The expanded section of the genome shows how evolution co-opted the organism's replication cycle and inserted logic to help perform the task.

This "early to bed, early to rise" behavior lets organisms finish tasks early during periods of resource availability, thereby increasing the probability of receiving a reward. It also helps avoid situations in which an organism starts working on a task but completes it just after the resource disappears, when there is no reward. Figure 5 shows a sample population that evolved this behavior, recorded in snapshots of a 60×60 grid during a single Avidian day.

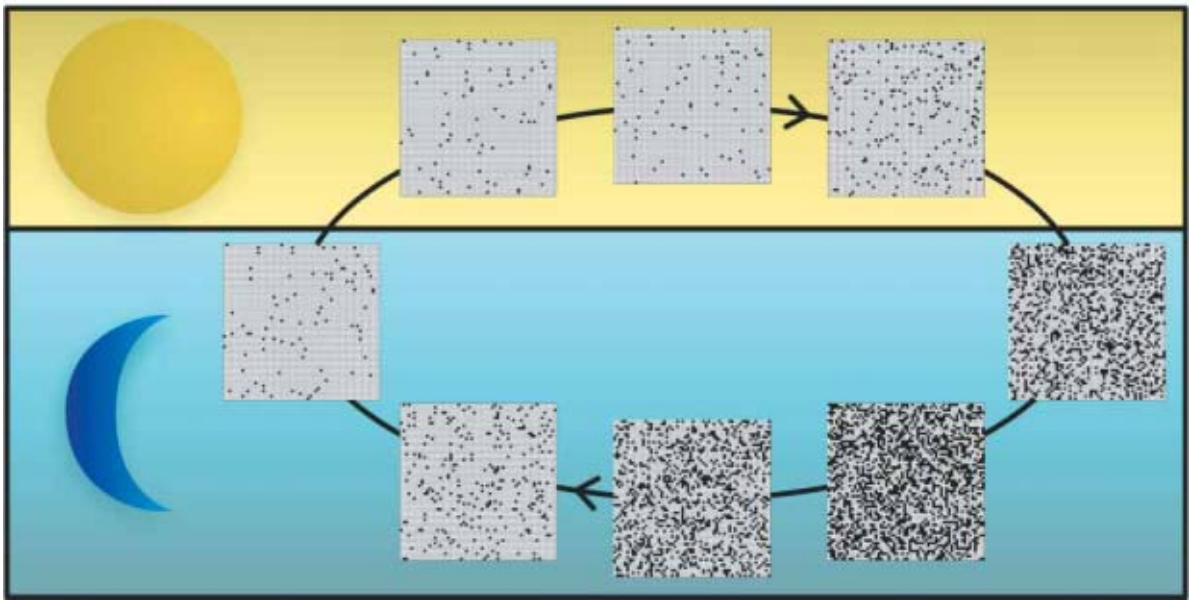


Figure 5. Representations of a population's response to the resource availability during an Avidian day. Black squares represent sleeping organisms, white squares represent awake ones. Snapshots in the figure's top half show the population when the resource is available, and they are rewarded for completing tasks. Snapshots in the figure's bottom half show the population when the resource is not available, and task completion goes unrewarded.

At this point in the run, the resource is available about 44 percent of the day. The black squares depict sleeping organisms, the white squares awake ones. The three snapshots at the top part of the figure depict the population's state when the resource is available, while the five snapshots on the bottom part depict the population's state when the resource is unavailable. This adaptive behavior arose in 37 out of 50 runs.

Our ongoing studies address conservation of energy balanced against other activities, such as detecting and reporting events of interest. We plan to test the most promising evolved solutions on sensor network simulators and, eventually, deploy them on physical devices and compare them to hand-built solutions.

Evolving behavioral models

We also use digital organisms to assist in constructing models of software behavior, including adding new functionality to an existing system. Software developers often use model-driven development (MDD)[8] to construct graphical models of desired structure and behavior, automatically transform the models into more formal specifications, and eventually generate the corresponding code.

Currently, many developers use the Unified Modeling Language to model systems. Despite MDD's many advantages, however, the construction of a UML behavioral model—which comprises a set of state diagrams for interacting objects—can be error-prone and difficult to automate, especially when extending an existing model to include new functionality. Digital evolution provides a means to generate possible solutions automatically.

Our approach, depicted in Figure 6, treats each Avidian as a generator of state diagrams: When the organism executes, it constructs an in-memory representation of one or more state diagrams. To implement this method, we extended Avida and integrated it with existing software engineering tools. First, we provide each organism with information about class diagram elements and, optionally, any existing state diagrams of the system. We call this information *instinctual knowledge*.

When replication creates a new organism, it is provided with a file containing its instinctual knowledge. For every class in the class diagram, the file contains an optional existing state diagram and lists of elements—such as triggers, guards, actions, and states. We also enhanced the Avida instruction set with instructions that let an organism use its instinctual knowledge to create additional transitions in one or more state diagrams.

For a given problem, the developer defines a collection of tasks that reward organisms for generating state diagrams that support scenarios, satisfy formally specified properties, and optimize software engineering metrics, such as minimizing the number of transitions in a diagram. To enable Avida to assess the completion of such tasks, we integrated it with a UML formalization framework, Hydra,[9] and the Spin model checker.[10] Hydra translates generated state diagrams into a representation in the Promela specification language, which Spin verifies against properties.

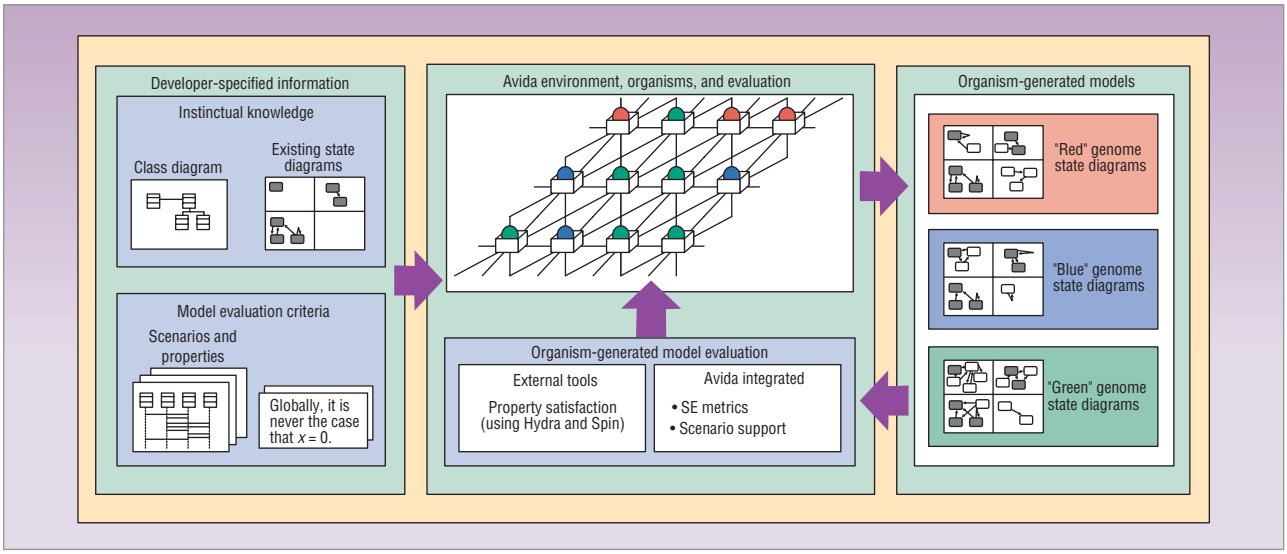


Figure 6. Using Avida to develop software state diagrams. Individual organisms are provided with instinctual knowledge of existing software and evolve to produce state diagrams that meet developer-specified requirements.

As in other Avida applications, a population starts with a single organism capable only of replication. As the organism and its descendants replicate, random mutations produce different genomes. Organisms that generate state diagrams exhibiting desired characteristics receive more CPU cycles and thus replicate faster.

Effectively, an Avida population is subject to a natural-selection pressure that rewards organisms for generating state diagrams that support key scenarios and satisfy critical properties. If an organism generates state diagrams that support all key scenarios and satisfy all properties, it has successfully and automatically generated a behavioral model for the system. We refer to the state diagrams that meet these requirements as *compliant* state diagrams. At this point, the experiment succeeds and we can halt it, or we might allow it to proceed to find other sets of compliant state diagrams. We used this approach to generate state diagrams describing new mobility behavior in a robot.[11] Researchers and developers can apply this technology to other domains exhibiting complex requirements.

FUTURE DIRECTIONS

Although evolutionary computation is a well-established computing subfield, we are just beginning to understand how to harness the evolution of self-replicating digital organisms. Several major lines of research offer opportunities for those interested in this area of study.

The first involves different architectures and instruction sets. Avida is an extensible platform, and various von Neumann CPU architectures have been implemented and used in past studies. Within the current Avida environment, we are investigating instruction sets with better support for flow control, function invocation, and context switching. However, fundamentally different computation models, such as data flow machines or even models based on processors found in natural systems, such as gene regulatory networks, might lead to the evolution of complex and adaptive behaviors.

We also plan to expand our work on evolving digital organisms to construct models of software and other aspects of computing systems. Integrating Avida with tools for automated software engineering helps address the increasing need for high-assurance, robust software that can tolerate adverse physical conditions and flaws in hardware fabrication. Moreover, Avidians can evolve to help design other structures—such as network topologies—important to distributed computing.

Mobility presents another major area of future study. Members of our group have recently modified Avida to let organisms move among cells, and we have started developing a continuous-space Avida environment in which the laws of physics govern movement and communication. We are particularly interested in the evolution of cooperative mobility control. Coordination of movements is critical to behaviors such as flocking, avoiding obstacles, and eluding enemies. Moreover, recent studies with mobile sensors have shown that it's possible to exploit mobility to provide certain benefits to network performance, energy conservation, and communication security. A fundamental question is whether digital evolution might find behaviors that enable a collection of mobile robots to adapt to, and perhaps exploit, current conditions in ways not otherwise apparent to human designers.

A related area of study involves integrating biomimetics and digital evolution. Evolution has produced complex behaviors in natural systems, which might provide an effective starting point for evolving control software for

Related Research

Research into harnessing evolution extends into both the design and behavior of virtual and embodied agents and machines. Evolution has been harnessed to create more realistic videogames, MEMS chips that operate under extreme conditions, and swarm behavior in robots. Research papers in this area can be found in journals and conferences sponsored by the IEEE Computational Intelligence Society, the ACM Special Interest Group for Genetic and Evolutionary Computation, and the International Society of Artificial Life, among others.

Table A displays a small sampling of the groups conducting research in this field. While these groups use widely varying underlying substrates, they all share the concept of harnessing evolution and using it to solve problems.

Table A. Sampling of groups applying evolutionary computing to systems design.

Laboratory/Group	Institution/Organization	Keywords
Neural Networks Research Group	University of Texas at Austin	Neuroevolution, self-organization, robotics, evolutionary computation
Dynamical and Evolutionary Machine Organization Laboratory	Brandeis University	Coevolution, evolutionary robotics, neuroevolution
Cornell Computational Synthesis Laboratory	Cornell University	Evolutionary robotics, modular robotics, rapid prototyping
IRIDIA Laboratory	Free University of Brussels	Swarm intelligence, swarm-bots, self-organizing systems, biological networks
Laboratory of Intelligent Systems	École Polytechnique Fédérale de Lausanne	Flying robots, artificial evolution, social systems
Adaptive Control and Evolvable Systems Group	US National Aeronautics and Space Administration	Automated design, system optimization
Digital Biology Interest Group	University College London	Evolutionary computation, bio-inspired computing, developmental systems
Evolutionary Computation Laboratory	University of Central Florida	Neuroevolution, coevolution, autonomous agents
Bionics and Evolutiontechnique Department	Technische Universität Berlin	Bio-inspired machines, bionics
Adaptive Computation Group	University of New Mexico	Artificial immune systems, genetic algorithms, biological modeling
Evolutionary and Adaptive Systems Group	University of Sussex	Artificial life, evolutionary computation, adaptive systems

robots. For example, some animal species exhibit fission-fusion relationships in which individuals join together for some tasks, such as guarding a den or attacking prey, but act independently at most other times. We can hand-code such behaviors in an Avida organism and use it to seed the evolutionary process. Evolution in Avida would likely modify the behaviors to account for differences between robots and animals, including both enhanced capabilities such as availability of radio communication and limitations such as physical agility.

Finally, we can explore the joint evolution of the system's morphology, or physical structure, and its control software. Several researchers use evolutionary computation to help design integrated software and hardware for robots.[12] After all, organisms' bodies and brains evolve together in nature. Indeed, some would argue that intelligent behavior can evolve only when the system's decision-making part is coupled with a physical body that has sensors and actuators. Others claim that the sense-and-respond functionality can be abstracted from the physical world (into software sensors and actuators, for example) and still lead to evolution of intelligent behavior. Using digital evolution, we have begun studies to help answer this question.

Our preliminary studies using Avida to evolve interesting behaviors show promise and open doors to several areas of future research. In addition, the "Related Research" sidebar profiles several other research groups that apply various forms of evolutionary computation to systems design. This problem domain appears to offer a fertile research area with potentially important implications, given the increasing complexity of computing systems. We hope this research community will continue to grow.

Further information on our research can be found at www.cse.msu.edu/thinktank. For papers on other digital evolution applications, and Avida downloads and accompanying documentation, see http://devolab.cse.msu.edu.

Acknowledgments

We gratefully acknowledge the contributions of the faculty and students in the Digital Evolution Laboratory at Michigan State University. This research was supported in part by the Michigan State University Quality Fund, the US Department of the Navy, Office of Naval Research under grant no. N00014-01-1-0744, the DARPA Fundamental Laws of Biology program, and National Science Foundation grants ITR-0313142, CCF-0523449, CCF-0541131, and CCF-0750787.

References

1. J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, 2003, pp. 41-50.
2. P.K. McKinley et al., "Composing Adaptive Software," *Computer*, July 2004, pp. 56-64.
3. D. Floreano, P. Husbands, and S. Nolfi, "Evolutionary Robotics," *Handbook of Robotics*, Springer-Verlag, 2008.
4. C. Adami, *Introduction to Artificial Life*, Springer-Verlag, 1998.
5. D.C. Dennett, "The New Replicators," *The Encyclopedia of Evolution*, M. Pagel, ed., vol. 1, Oxford Univ. Press, 2002, pp. E83-E92.
6. R.E. Lenski et al., "The Evolutionary Origin of Complex Features," *Nature*, vol. 423, 2003, pp. 139-144.
7. B. Beckmann, P.K. McKinley, and C.A. Ofria, "Evolution of Adaptive Sleep Response in Digital Organisms," *Proc. 9th European Conf. Artificial Life*, Springer, 2007, pp. 233-242.
8. D.C. Schmidt, "Model-Driven Engineering," *Computer*, Feb. 2006, pp. 25-31.
9. W.E. McUmber and B.H.C. Cheng, "A General Framework for Formalizing UML with Formal Languages," *Proc. IEEE Int'l Conf. Software Eng.* (ICSE 01), IEEE Press, May 2001, pp. 433-442.
10. G. Holzmann, *The Spin Model Checker, Primer and Reference Manual*, Addison-Wesley, 2004.
11. H.J. Goldsby et al., *Automatic Generation of UML Behavioral Models through Digital Evolution*, tech. report MSU-CSE-07-194, Dept. Computer Science and Eng., Michigan State University, East Lansing, Mich., 2007.
12. H. Lipson, "Evolutionary Robotics and Open-Ended Design Automation," *Biomimetics*, B. Cohen, ed., CRC Press, 2005, pp. 129-155.

Philip McKinley *is a professor in the Department of Computer Science and Engineering at Michigan State University, East Lansing, Michigan. His research interests include self-adaptive software, autonomic computing, and digital evolution. McKinley received a PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the IEEE and the ACM. Contact him at mckinley@cse.msu.edu.*

Betty H.C. Cheng *is a professor in the Department of Computer Science and Engineering at Michigan State University. Her research interests include model-driven engineering, formal and automated analysis of high-assurance systems, and adaptive and autonomic systems. Cheng received a PhD in computer science from the University of Illinois at Urbana-Champaign. She is a senior member of the IEEE. Contact her at chengb@cse.msu.edu.*

Charles Ofria *is an assistant professor in the Department of Computer Science and Engineering and the Ecology, Evolutionary Biology, and Behavior Program at Michigan State University. His research interests include digital evolution, biocomplexity, and bioinformatics. Ofria received a PhD in computation and neural systems from the California Institute of Technology. Contact him at ofria@cse.msu.edu.*

David Knoester *is a doctoral student in the Department of Computer Science and Engineering at Michigan State University. His research interests include digital and biological evolution, self-organizing systems, and distributed computing systems. Knoester received an MS in computer science from Michigan State University. He is a member of the IEEE and the ACM. Contact him at dk@cse.msu.edu.*

Benjamin Beckmann *is a doctoral student in the Department of Computer Science and Engineering at Michigan State University. His research interests include sensor networks, autonomic computing, and evolutionary robotics. Beckmann received an MS in computer science from Western Michigan University. He is a member of the IEEE and the ACM. Contact him at beckma24@msu.edu.*

Heather Goldsby *is a doctoral student in the Department of Computer Science and Engineering at Michigan State University. Her research interests include model-driven development of high-assurance systems, dynamically adaptive systems, and using digital evolution to support software development. Goldsby received an MS in computer science from Michigan State University. She is a member of the IEEE and the ACM. Contact her at hjg@cse.msu.edu.*

Mining the Social Fabric of Archaic Urban Centers with Cultural Algorithms

Robert G. Reynolds, Mostafa Ali, and Thaer Jayyousi
Wayne State University

The authors use decision trees to characterize location decisions made by early inhabitants at Monte Albán, a prehistoric urban center, and inject these rules into a socially motivated learning system based on cultural algorithms. They can then infer an emerging social fabric whose networks provide support for certain theories about urban site formation.

> Culture change in the direction of increased scale and complexity can occur in varied ways. I suggest that the cultural ecologists should do as others have and view this variety as a source of stimulation for theory-building.
>
> —R.E. Blanton[1]

Applying a suite of tools from artificial intelligence and data mining to existing archaeological data from Monte Albán, a prehistoric urban center, offers the potential for building agent-based models of emergent ancient urban centers.

Specifically, we examine the period of occupation associated with the emergence of this early site. Our goal is to generate a set of decision rules using data-mining techniques and then use the cultural algorithm toolkit (CAT) to express the underlying social interaction between the initial inhabitants.

MONTE ALBÁN: AN ARCHAIC MESOAMERICAN CITY

The archaeological site of Monte Albán is situated in the Valley of Oaxaca, located in central Mexico. Figure 1 shows the site's basic physical layout, with its central plaza located to the south of the flat hilltop at 400 meters above the valley floor. On the site are more than 2,000 terraces where site occupants lived and worked.

The Tierras Largas period (1400-1150 BC) marks the beginning of early village settlement in this valley, followed by three more periods of social evolution, before the state emerged at Monte Albán in the Monte Albán Ia period (500-300 BC). The valley came under control of the state by Monte Albán II (150-100 BC to 200 AD), and Monte Albán IIIa (200-500 AD) signaled the decline of the state and its succession by a collection of city-states localized in different parts of the valley. The phases described here represent uneven slices through time and are defined in terms of the pottery found at Monte Albán. Each phase represents a change in the predominant style of pottery. The chronology of the phases was determined by radiocarbon dating the pottery of each style.[1,2]

MONTE ALBÁN DATA SET

Each terrace at Monte Albán is described in terms of hundreds of cultural and environmental variables, which come from an intensive archaeological survey undertaken as part of the Valley of Oaxaca Settlement Pattern Project.[1,2] Our study applied several different data-mining techniques including decision-tree learning to a data set consisting of the more than 2,000 hillside terraces that make up the Monte Albán occupation.

Hypothetical models of urban growth

Cities such as Monte Albán are one of the most "spectacular settlement types of pre-Columbian Mesoamer-

Published by the IEEE Computer Society

ica."[3] These prehistoric urban centers were both the product of social changes and the platform on which future changes would take place. These archaic cities can be described in terms of the following dimensions:

- size, either of the population or the area covered by the city;
- location or environment (on a river, a coast, a mountaintop, or another location);
- function (ceremonial, commercial, defensive, or administrative);
- position in a settlement hierarchy; and
- morphology, or form, as influenced by all the above.

The basic framework for our approach revolves around the morphology because it reflects all variables.

Further, Joyce Marcus stated that a city contains four basic parts[4]:

- homogeneous parts such as large multifamily living quarters;
- central part or plaza;
- circulatory part or road system; and
- special area such as a marketplace.

Not all of these parts are necessarily found in all cities, but they reflect features that are likely to be planned parts of the city's morphology or shape.

A city's morphology also has unplanned components that reflect its growth or decline over time. Researchers have developed several models within the past half century to describe these emergent shapes in terms of modern cities. These models include the following:

- concentric zones based upon the growth of modern cities, with a city center surrounded by concentric zones of activity;
- sectors in which differences in land use near the center are preserved and "fan out" like slices of pie as the city grows; these differences can reflect residential, economic, and civic ceremonial differences;
- multiple nuclei, an extension of the above in which there are multiple nuclei around which the activities are organized, either as sectors or concentric zones of activity (city center, light manufacturing, business, and low-, middle- and high-status residences, and so forth).[3]

Given the above models, it is clear that within a modern city there will be patterns that reflect the structure of the city as a whole, as well as patterns that reflect specific areas of localized activity within the city. We expect that our data-mining activities, if successful, will produce rules that pertain to settlement patterns over the archaic city, while others will pertain to specific regions. The spatial expression of these extracted rules might allow us to identify the particular morphology associated with a given city. Our goal will be to see if we can use any of these models to describe the morphology of the archaic site.

Figure 1. The Monte Albán hilltop site.

Using decision trees to model site settlement decisions

A natural vehicle for the expression of the decision-making rules is the decision tree. We have used decision trees previously to characterize settlement patterns in the area around Monte Albán over a single time period,[5,6] but the urban area under study here is much more complex than for the smaller peripheral settlements. A decision tree is a directed acyclic graph that describes a pattern in terms of choices relative to selected variables. Variable choices made at the top of the tree are more important in defining the pattern than those made near the bottom. A path from the root node to a leaf node in a decision tree represents a sequence of actions or choices.

The decision-tree structure is particularly appropriate here because we expect a certain hierarchical structure in how the variables influence a decision given that site formation in all of the models described above occurred relative to a certain component—for example, the main plaza or a road system.

MODELING THE SOCIAL FABRIC WITH CULTURAL ALGORITHMS: THE CAT SYSTEM

Cultural algorithms are one of several approaches to modeling the use of social intelligence to solve problems in optimization, including particle swarm optimization and ant-colony optimization.[7,8] The CA is a class of computational models derived from observing the cultural evolution process in nature.[9] It has three major components: a population space, a belief space, and a protocol that describes how the first two components exchange knowledge. The population space can support any population-based com-

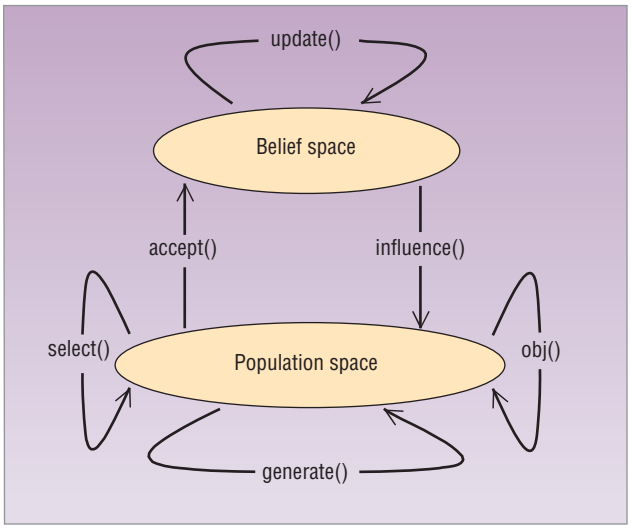


Figure 2. The cultural algorithm. The CA has three major components: a population space, a belief space, and a protocol that describes how the first two components exchange knowledge. The population space can support any population-based computational model, such as genetic algorithms or evolutionary programming.

putational model, such as genetic algorithms or evolutionary programming. The basic framework is shown in Figure 2.

A CA is a dual inheritance system that characterizes evolution in human culture at both the macroevolutionary level, which takes place within the belief space, and at the microevolutionary level, which occurs in the population space. Knowledge produced in the population space at the microevolutionary level is selectively accepted or passed to the belief space and used to adjust the knowledge structures there. This knowledge can then be used to influence the changes that the population makes in the next generation.

We have embedded the CA framework within the Repast agent-based simulation system.[10] The resulting system is called the cultural algorithm toolkit (CAT). The system is composed of the cultural algorithms, the core system, the visualization subsystem, a database (Access or MySQL), and the analyzer subsystem (geometry and statistics). We use the visualization subsystem to display the results, draw some conclusions about the data, and construct some basic real-time dynamic charts and statistics.

Within the CA core system are several components: the belief space, the population, the acceptance function, and the influence function. In the CAT system, we implement each component as simply as possible. This allows us to add complexity into the system incrementally.

Five default knowledge sources

In our CAT implementation there are five default knowledge sources, each of which can be used in some form of socially motivated problem solving. Also, there is a basis for the presence of each knowledge type in prehuman species as well. Thus, it isn't necessary to view the knowledge sources as unique to humans.

Topographical knowledge was originally proposed to reason about region-based functional landscape patterns.[11] It can potentially distribute individuals over the entire landscape. It was motivated in conjunction with data-mining problems where the problem space was so large that a systematic way of partitioning the space during the search process was needed. In our problem, the topography relates to the hilltop at Monte Albán. It is clear that topographic factors are influential in terms of settlement here in a number of ways. For example, topography will influence the location of the central plaza and the main roads. Access to both of these features could be important factors in choosing a terrace location.

Normative knowledge is a set of promising variable ranges that provide standards for individual behaviors and guidelines within which individual adjustments can be made. Normative knowledge came into play during the learning of rules for expert system applications. Normative knowledge directs individuals to "jump into the good range" if they are not already there. Here, the norms relate to the ranges of values associated with the extracted rules for the site. Each rule has a domain of values for the variables that it uses explicitly. We allow for some movement outside of those ranges probabilistically.

Domain knowledge employs knowledge about the problem domain to guide search. Saleh Saleem first used domain knowledge to guide the search for resource cones of maximum height in a landscape.[12] Here, the domain knowledge relates to the set of location rules that are generated in the data-mining process. In the example we use, these rules work to direct individuals into specific areas of the hilltop site. However, we can use other rules relating to how the situated terraces are used in the site.

Situational knowledge provides a set of exemplary cases that are useful for the interpretation of specific individual experiences. Situational knowledge leads individuals to "move toward the exemplars." This was the earliest knowledge source used with the CA and was inspired by elitist approaches in genetic algorithms. This knowledge source collaborates with domain knowledge to exploit above-average regions. In our example, exemplars can correspond to terraces that are positioned in strategic locations on the hilltop relative to important features of the site. For example, some regions of the site contain quarryable stone. Access to this resource is important since the steep roads make carrying large pieces of stone to the site difficult.

Historical or temporal knowledge monitors the search process and records important events in the search. Saleem first used this knowledge source,[12] which

Bin Peng expanded on.[13] Individuals guided by historical knowledge can consult recorded events for guidance in predicting a good direction to move in. In this example, we focus on the initial phase of occupation for the site, and we assume no prior history of occupation. However, simulation of successive phases will require additional use of this knowledge. For example, once a terrace is cleared for use in one phase of occupation, it becomes part of the "built environment." This built environment can certainly condition the selection of new terraces in the future.

Population model

The population model used here is a simple evolutionary algorithm in which each individual corresponds to a potential terrace occupant or family. Each of the occupants will have a set of background variables that indicate social status, where they came from, terrace utilization goals, and so forth. We support two basic population models for our agents in the CAT system, genetic algorithms and evolutionary programming.[14,15]

The CAT has two basic experimental environments. The first is the spatially motivated problem-solving environment in which agents move through a potentially complex two-dimensional space. We call this the resource "cones world" problem, and the default algorithm for this is genetic algorithms. The second experimental environment uses evolutionary programming as the default algorithm to support real-valued, function-optimization problems relating to engineering design. In this particular application, we use the cones world framework to represent the problem space.

Acceptance function

The acceptance function determines which individuals and their behaviors can impact the belief space. It is often specified as a percentage of the number of current individuals ranging between 1 percent and 100 percent of the population size, based upon selected parameters such as performance. For example, we can select the best performers (top 10 percent), worst performers (bottom 10 percent), or any combination. In our example, we are interested in the settlement decisions of all newcomers to the site.

MVT influence function

The basic problem-solving engine that we use in the CAT implements Charnov's marginal value theorem in metaphorical form.[16] From this perspective, each knowledge source is viewed as a predator that explores a portion of the function landscape. That portion is its patch. In our model, the patch associated with a predator is a bounding box that is centered at the average value for the individuals that it generates, and it is one standard deviation from the average in each of the dimensions for the problem.

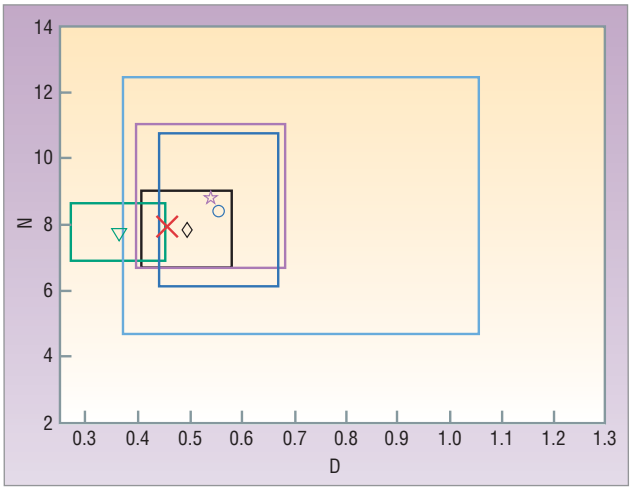


Figure 3. The knowledge swarming process. The process is described statistically through the distribution of the bounding boxes for each of the five knowledge sources. Black represents situational knowledge, and turquoise represents topographical knowledge;* N *and* D *represent dimensions for a spring optimization problem.

In foraging theory, Charnov showed that under certain conditions the marginal value theorem could optimize the long-term average rate of energy intake within a patch-based environment.[16] The principle behind the marginal value theorem is that residence time in a patch by a forager affects the expected energy gain. The marginal value principle states that the forager should reside in the patch "until the intake rate in a patch drops to the average rate for the habitat … it is the 'moving-on threshold' intake rate that is important." When doing so, the forager will maximize the individual's average long-term energy intake.

One key assumption is that the gain function associated with a patch initially increases but eventually accelerates negatively. This works here because of the difference in temporal and spatial granularity of the knowledge sources. As the exploiter knowledge sources move into an area initially found by an explorer, they will exploit the space to a finer degree and a higher density. Thus, they will have more high-performing individuals and improve the overall average. On the other hand, the explorers' performance relative to the overall average will go down. This will cause their operators to take larger steps, and any improvement will carry substantial weight, causing a shift in their bounding boxes in that direction.

The result is a "knowledge swarming" process in which the bounding boxes associated with the knowledge sources move in a synchronized way over the landscape. In Figure 3, the bounding boxes are color-coded relative to their knowledge source. The situational knowledge source (black) is focused around the best point, while topographic knowledge (turquoise) has begun to move

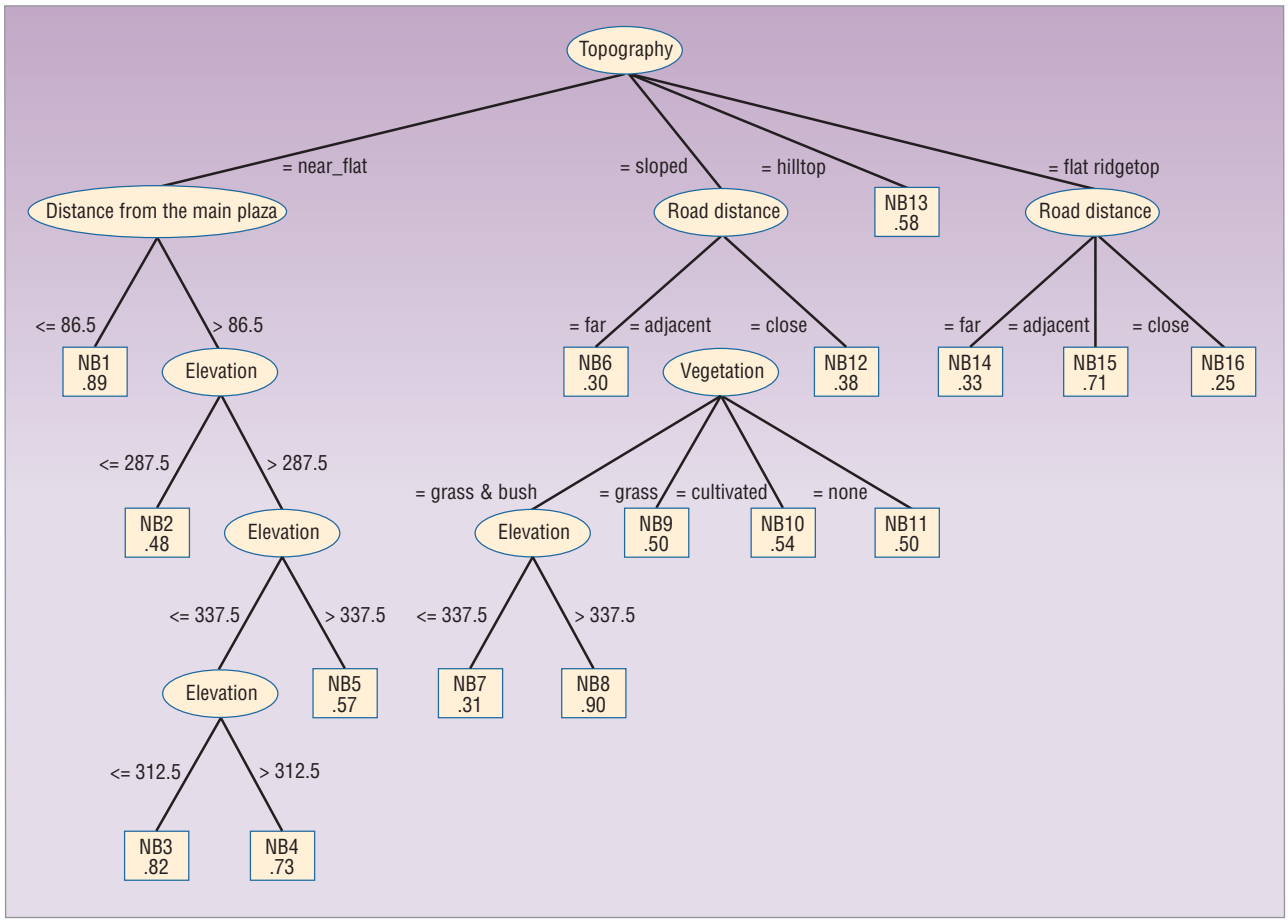


Figure 4. The Naïve Bayesian decision tree produced to predict the 390 occupied terraces in the initial phases of occupation of the site. Each path through the tree corresponds to a rule. The rules are numbered based upon a preorder traversal of the tree. The NB leaf nodes contain the prior probability that a terrace will be occupied given the associated rule.

away from the area relative to the current two-dimensional problem space. *N* and *D* represent dimensions for a spring optimization problem.[13] Here, the swarming activity results from the complementary granularity of the knowledge sources and provides a systematic way of exploring the environment.

Social fabric

From a theoretical perspective, we view individuals in the real world as participating in a variety of different networks. A population can support several layers of such networks. The interplay of these various network computations is designated as the "social fabric." Although this notion of social fabric has appeared metaphorically in various ways within computer science, here we view it as a computational tool that influences the action and interaction of the various knowledge sources.

The rules in our belief space are expressed through the networks of individuals they influence. Informally, we have *N* networks and *M* individuals. An individual can be associated with one or more networks. In a given network, only certain information is allowed to flow between nodes. We can view each network as being produced by a single thread that links up the participating nodes. Information, matter, and energy can flow along the thread. A set of connector threads is used to weave the threads together to produce the social fabric. They do this by associating a node in one network with a node in another such that flow in one network can constrain flows in the other and vice versa. The number of connections between the networks reflects the tightness of the weave. If the weave is too tight, there is less flexibility when the system is placed under stress.

Researchers have proposed many different network topologies to connect individuals together. For example, in the gbest topology, each individual is connected to all other individuals. In lbest, each individual is connected to only two other individuals. In the FourSquare communication topology, each individual is connected to four others only. In the CAT system, we support a subset of these topologies. Here, we use the FourSquare topology and assume that an occupant is connected to four other occupants.

DATA-MINED DECISION RULES

While we can generate decision trees for many different aspects of the site, we focus on basic site-location deci-

sions. These decisions are expressed in terms of a subset of environmental variables that were determined, after preliminary screening, to potentially impact terrace location decisions at the site. We selected Kohavi's Naïve Bayes decision-tree algorithm, NBTree, implemented in the WEKA data-mining toolkit because it scaled up with increasing sample size.[17] In period Ia, only 390 terraces were occupied, but in later periods around 2,000 terraces could be occupied, and it is important that the decision trees be comparable for samples of different sizes. Also, since many of the variables are not measured in precise terms, there is some room for variability. This variability can be expressed in the decision tree's Bayesian component.

Table 1. Each of the 16 rules along with the probability of classifying the related terraces as occupied or not. Only two of the 16 nodes have no decided majority for one class or the other.

Naïve Bayes leaf number	Prior probability Yes	Prior probability No	Topography	Number of associated terraces
1	.89	.11	near_flat & Distance from Mean Plaza <= 86.5	110
2	.48	.52	near_flat & Distance from Mean Plaza > 86.5 & Elevation<= 287.5	11
3	.82	.18	near_flat & Distance from Mean Plaza > 86.5 & Elevation> 287.5 & Elevation <= 337.5 & Elevation <= 312.5	22
4	.73	.27	near_flat & Distance from Mean Plaza > 86.5 & Elevation> 287.5 & Elevation <= 337.5 & Elevation > 312.5	7
5	.57	.43	near_flat & Distance from Mean Plaza > 86.5 & Elevation> 287.5 & Elevation > 337.5	11
6	.30	.70	sloped & Road Distance= far	54
7	.31	.69	sloped & Road Distance= directly_adjacent & vegetation=grass_and_brush & Elevation <= 337.5	8
8	.90	.10	sloped & Road Distance= directly_adjacent & vegetation=grass_and_brush & Elevation > 337.5	26
9	.50	.50	sloped & Road Distance= directly_adjacent & vegetation=grass_only	0
10	.54	.46	sloped & Road Distance= directly_adjacent & vegetation=cultivated	6
11	.50	.50	sloped & Road Distance= directly_adjacent & vegetation=none	0
12	.38	.62	sloped & Road Distance= close	109
13	.58	.42	hilltop	6
14	.33	.67	flat_ridgetop & Road Distance= far	0
15	.71	.29	flat_ridgetop & Road Distance= directly_adjacent	19
16	.25	.75	flat_ridgetop & Road Distance= close	1

Naïve Bayes decision-tree approach

NBTree is a hybrid algorithm that combines both decision-tree and Naïve Bayes classifiers; the result is a decision tree of nodes and branches with Bayesian classifiers at the leaf nodes.

Given a node associated with a set of instances, the algorithm builds the model from the top down. For a given node, a number of positive and negative examples are associated with it. The node's utility is computed by discretizing the data and performing a fivefold cross-validation to estimate the current accuracy using Naïve Bayes. Each of the possible attributes is evaluated to see if it can produce a split of the data such that the weighted sum of the utility of the nodes produced by the split is significantly better than that of the current node. Significantly better means a minimum amount of error reduction and that the resulting nodes each have a specified minimum number of individuals. For a given variable, the nodes that result from its split are produced by entropy minimization for continuous variables, and by associating a specific class with each node for discrete variables.

A location decision tree for Monte Albán

Figure 4 shows the decision tree we generated for Monte Albán to predict whether a terrace was occupied in period Ia or not based on selected environmental variables. The leaf nodes are Naïve Bayes classifiers for the terraces associated with each rule, where a rule is a path from the root to a leaf node. Table 1 shows each of the 16 rules along with the probability of classifying the related terraces as occupied or not. Only two of the 16 nodes have no decided majority for one class or the other.

Notice that the most important factor is topography. If the topography is nearly flat, the distance of the terrace to the main plaza area at the south end of the hilltop is critical. If it is close, it is likely to be occupied. On the

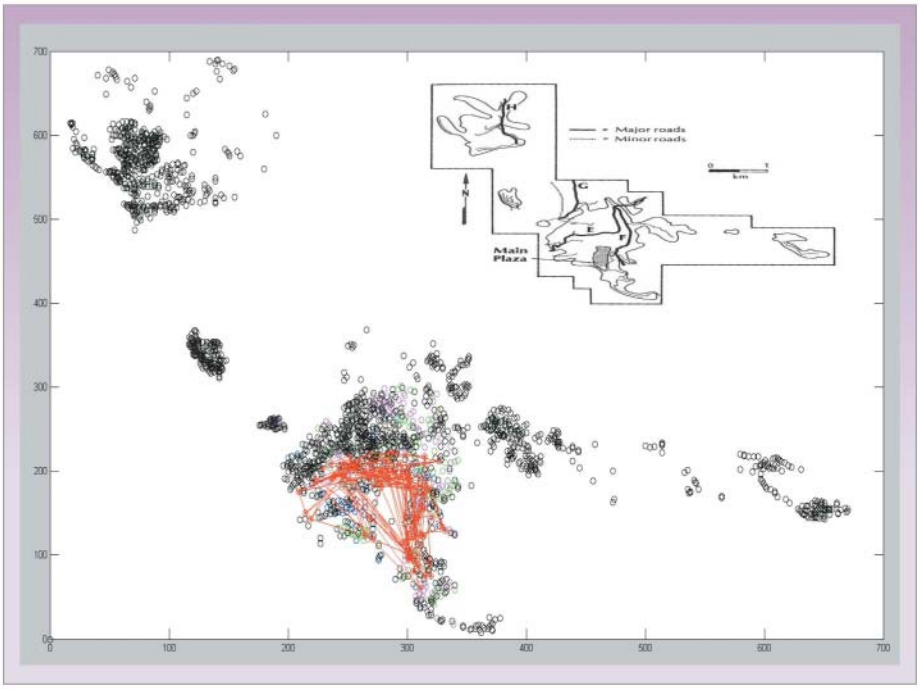


Figure 5. The social fabric as woven by the three example rules, 1 (red), 6 (blue), and 10 (green). The empty white area near the bottom is the central plaza. The red region surrounding the central plaza at the northern end corresponds to the primary elite and nonelite residential areas at the site. The strong vertical set of links on the eastern side of the site relate to terraces adjacent to one of the two major roads at the site as shown in the inset map.

other hand, if the topography is not nearly flat, proximity to the road comes into play. If a terrace is on sloped terrain and far from a road, it is not likely to be occupied. If it is adjacent to the road and on soil that can grow both grass and brush and at relatively high elevation, it is likely to be occupied. The vegetation variable is a surrogate for how much slope there is. The steeper the slope, the less likely that vegetation will be there.

The decision-tree classification scheme accurately predicted 815 of the terraces correctly, with a true prediction rate of 89 percent for occupied terraces and a somewhat lower rate, 72 percent, for the unoccupied terraces. The key problem here is that the rules classified several unoccupied terraces as occupied. This is interesting because in the early phase of occupation, there are many more available terraces than there are individuals to occupy them. So, in fact, the algorithm is suggesting that if more people were interested in occupying the site, these terraces might have been occupied as well.

Cultural algorithms

Now we can use the rules extracted from the database to direct the occupation of the site in a probabilistic fashion in an initial year of occupation. Each rule resides in the database and selects an individual from the population to control.

The likelihood of controlling an individual initially is a function of the size of the set associated with the Naïve Bayes classifier for that rule. Individuals are initially placed into a FourSquare topology where each is connected to four other individuals. Each knowledge source is assigned a region of a roulette wheel based on its overall influence or probability. The roulette wheel is spun, and the selected knowledge source (rule) selects an individual. Next, the knowledge source randomly selects a currently unoccupied terrace and determines whether its variables fit the rule conditions. If it does, the individual is assigned to that terrace. If it does not, the knowledge source spins the wheel again. This way, the most predominant rules will proceed to fill in the terraces first, and lesser rules will fill in the vacant slots.

Since there are 390 occupied terraces, we assign 390 individuals to terraces. Of course, some of the terraces to which they are assigned might not actually be occupied because a rule can randomly select a terrace that matches its condition but which is not occupied. Since there are more than 2,000 surveyed terraces, there might be many other terraces that have the same properties but are not occupied yet. The accuracy of each rule in placing individuals on terraces is then fed back to the belief space to adjust the rule conditions, then the initialization year can be simulated again. The goal is to gradually adjust the rules in context to produce increased accuracy. In this article, we are just describing the results of such a first pass.

With a given placement of individuals, we can draw arcs between individuals that are connected together in the topology and are assigned to cells using the same knowledge source. This corresponds to the notion that individuals who were linked to each other prior to arrival and settle in similar circumstances will continue to be linked. In this way, we are using the social fabric that binds individuals together to reflect the interaction between individuals residing in terraces of similar type. There are, of course, many different ways of interpreting this social fabric. However, our aim here is to demonstrate the insights that selecting even a simple fabric can provide, even after just one generation of assignments.

In Figure 5 we give the networks associated with individuals controlled by the three of the rules, 1, 6,

and 10, as examples of different aspects of site occupation that relate directly to our previous models of urban site formation. Rule 1 is coded as red, 6 as blue, and 10 as green. If we examine the figure in more detail, we see an empty white rectangular area near the bottom. This is where the central plaza is located. Around the plaza are terraces. Terraces that are occupied and associated with a given rule are colored accordingly. Links between connected individuals residing in the same type of terrace are colored the same as the nodes. Terraces occupied in the phase but not associated with any of the rules shown here are in pink. Terraces that are not occupied during this period are etched in black.

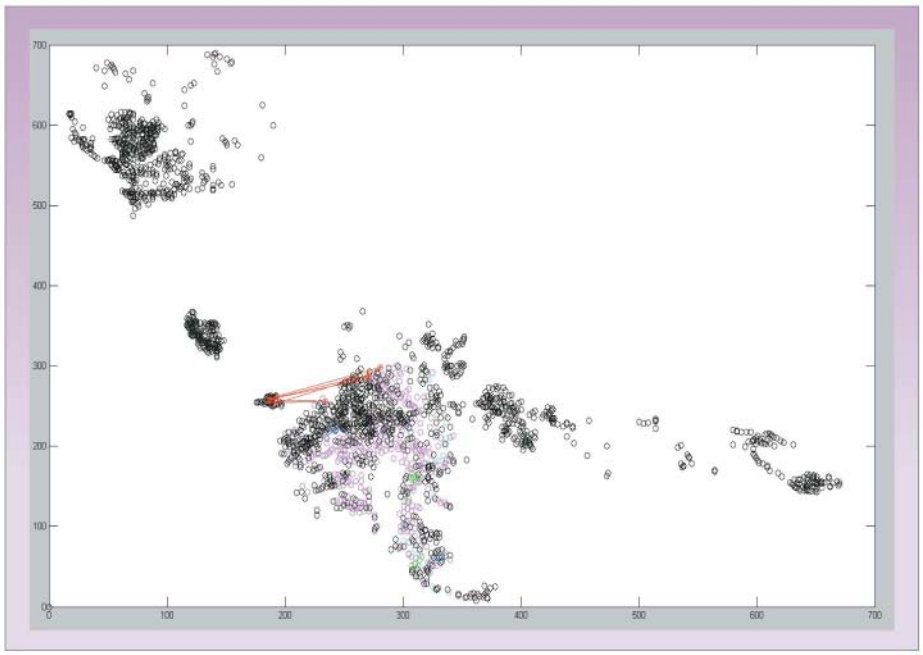


Figure 6. The social fabric woven by rule 2. The pink sites are occupied but not associated with this rule. The links suggest a relationship between the outpost at El Mogotillo and the main entrance points on the western part of the site.

Notice that the red region surrounds the central plaza at the northern end like a horseshoe. Many threads are visible there. That area corresponds to the primary elite and non-elite residential areas at the site. This resembles a central place model of occupation and suggests that residential occupation initially is strongly oriented around the central plaza.

On the other hand, we can observe a strong vertical set of links associated with rule 10 on the eastern side of the site. These links relate to terraces adjacent to one of the two major roads at the site. The fact that the other road did not show up as strongly suggests that this might have been the first major established route. If people were immigrating from the north, which at the time was the most populous part of the valley, this would be a logical entry route for them. The inset in Figure 5 is a map of the known ancient roads at the site. Since we can't determine from the data itself which roads were developed first, the emergent social fabric suggested that the north-south route was the earliest.

Figure 6 gives another interesting insight. Rule 2 in red gives a set of links between El Mogotillo, a small hill just west of Monte Albán, and terraces on the western periphery near the major entrances to the site, suggesting that there was some relationship between those terraces relative to restricting or controlling site access. On the map in Figure 5, the hill is located at a confluence of routes that relate to the western entrance of the city. This relationship is certainly not evident from the original data set alone.

The integration of data-mining and agent-based social learning tools allows us to infer patterns of social interaction at the site. In particular, we used decision trees to characterize terrace location decisions made by the early inhabitants of the major archaic urban center at Monte Albán. We then injected these rules into a socially motivated learning system based on cultural algorithms. The result was the expression of these location decisions within an inferred social fabric.

The resulting example rules and their networks provide support for two urban models. There appears to be some evidence for a strong residential focus around the area that became the central plaza, which supports the central place concept. The site's physical structure precludes a concentric circle, but a good proportion of that is visible in the emergent network. Also, evidence for the organization of terraces around at least one of the major roads at the site suggests that it was one of the first to be developed. Although these results are preliminary in nature, further work will focus on determining the differences between the terraces associated with each of the rules.

We have just scratched the surface by focusing on an initial pass through the system. Doing so reveals the interesting social structures that lie just beneath the surface of the extracted data. In future work, we will experiment with how well the system can adjust the rules collectively to better predict terrace occupation and observe the corresponding changes in the social fabric as that happens.

Acknowledgment

This research was supported by NSF Integrative Graduate Education and Research Training grant no. 0654014.

References

1. R.E. Blanton, *Monte Albán: Settlement Patterns at the Ancient Zapotec Capital*, Academic Press, 1978.
2. R.E. Blanton et al., *Monte Albán's Hinterland, Part I, the Prehispanic Settlement Patterns of the Central and Southern Parts of the Valley of Oaxaca, Mexico*, The Regents of the Univ. of Michigan, The Museum of Anthropology, 1982.
3. J. Marcus and K.V. Flannery, *Zapotec Civilization: How Urban Societies Evolved in Mexico's Oaxaca Valley*, Thames and Hudson, 1996.
4. J. Marcus, "On the Nature of the Mesoamerican City," *Prehistoric Settlement Patterns: Essays in Honor of Gordon R. Willey*, E. Z. Vogt and R. M. Leventhal, eds., Univ. of New Mexico Press, 1983, pp. 195-242.
5. R.G. Reynolds and H. Al-Shehri, "The Use of Cultural Algorithms with Evolutionary Programming to Guide Decision Tree Induction in Large Databases," *Proc. IEEE World Congress on Computational Intelligence*, IEEE Press, 1998, pp. 541-546.
6. R.G. Reynolds, "The Impact of Raiding on Settlement Patterns in the Northern Valley of Oaxaca: An Approach Using Decision Trees," *Dynamics in Human and Primate Societies*, T. Kohler and G. Gumerman, eds., Oxford Univ. Press, 1999.
7. J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," *Proc. IEEE Int'l Conf. Neural Networks*, IEEE Press, 1995, pp. 12-13.
8. M. Dorigo, V. Maniezzo, and A. Colorni, "Ant System: Optimization by a Colony of Cooperating Agents," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 26, no. 1, 1996, pp. 29-41.
9. R.G. Reynolds, "An Introduction to Cultural Algorithms," *Proc. 3rd Ann. Conf. Evolutionary Programming*, A.V. Sebald and L.J. Fogel, eds., World Scientific Publishing, 1994, pp. 131-139.
10. M.J. North, N.T. Collier, and J.R. Vos, "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit," *ACM Trans. Modeling and Computer Simulation*, ACM Press, vol. 16, no. 1, 2006, pp. 1-25.
11. X. Jin and R.G. Reynolds, "Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: A Cultural Algorithm Approach," *Proc. IEEE Congress on Evolutionary Computation*, IEEE Press, 1999, pp. 1672-1678.
12. R.G. Reynolds and S.M. Saleem, "The Impact of Environmental Dynamics on Cultural Emergence," *Perspectives on Adaptions in Natural and Artificial Systems*, Oxford Univ. Press, 2001, pp. 253-280.
13. R.G. Reynolds and B. Peng, "Knowledge Learning and Social Swarms in Cultural Algorithms," *J. Mathematical Sociology*, vol. 29, 2005, pp. 1-18.
14. J.H. Holland, *Adaptation in Natural and Artificial Systems*, Univ. of Michigan Press, 1975.
15. D.B. Fogel, *Evolutionary Computation—Toward a New Philosophy of Machine Learning*, IEEE Press, 1995.
16. E.L. Charnov, "Optimal Foraging: The Marginal Value Theorem," *Theoretical Population Biology*, vol. 9, 1976, pp. 129-136.
17. R. Kohavi, "Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid," *Proc. 2nd Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, 1996, pp. 202-207.

Robert G. Reynolds *is a professor of computer science at Wayne State University and an associate research scientist in the Museum of Anthropology at the University of Michigan-Ann Arbor. His research interests include artificial intelligence, evolutionary computation, and cultural algorithms. Reynolds received a PhD in computer science from the University of Michigan-Ann Arbor. He is a member of the IEEE, the IEEE Computer Society, the ACM, the American Association of Artificial Intelligence, the Evolutionary Programming Society, the American Association for the Advancement of Science, and the Society for American Archaeology. Contact him at reynolds@cs.wayne.edu.*

Mostafa Ali *is a PhD candidate in the Computer Science Department at Wayne State University. His research interests are cultural algorithms, expert systems, and knowledge-based software engineering. Ali received an MSc in computer science from the University of Michigan-Dearborn. He is a member of the IEEE, the IEEE Computer Society, and the ACM. Contact him at mostafa@wayne.edu.*

Thaer Jayyousi *is an MS candidate in the Department of Computer Science at Wayne State University. His research interests include artificial intelligence, data mining, and machine learning. Jayyousi received a BS in computer science from Wayne State University. Contact him at al6854@wayne.edu.*

JANUARY 2008

Advertisers	Page Number
Cornell University	75
Hong Kong Baptist University	76
IEEE Computer Society Digital Library	Cover 2
IEEE Computer Society Membership	94-96
InfoVis 2008	Cover 4
Microsoft	27
National Sun-Yat Univesity, Taiwan	79
The Technical University of Denmark	77
University of Louisville	74
VIS 2008	Cover 4
Classified Advertising	*74-81*

Boldface denotes advertisements in this issue.

Computer
IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, California 90720-1314
USA
Phone: +1 714 821 8380
Fax: +1 714 821 4010
http://www.computer.org
advertising@computer.org

Advertising Sales Representatives

Mid Atlantic (product/recruitment)
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0164
Email: db.ieeemedia@ieee.org

New England (product)
Jody Estabrook
Phone: +1 978 244 0192
Fax: +1 978 244 0103
Email: je.ieeemedia@ieee.org

New England (recruitment)
John Restchack
Phone: +1 212 419 7578
Fax: +1 212 419 7589
Email: j.restchack@ieee.org

Northwest (product)
Lori Kehoe
Phone: +1 650-458-3051
Fax: +1 650 458 3052
Email: l.kehoe@ieee.org

Southeast (recruitment)
Thomas M. Flynn
Phone: +1 770 645 2944
Fax: +1 770 993 4423
Email: flynntom@mindspring.com

Midwest (product)
Dave Jones
Phone: +1 708 442 5633
Fax: +1 708 442 7620
Email: dj.ieeemedia@ieee.org

Will Hamilton
Phone: +1 269 381 2156
Fax: +1 269 381 2556
Email: wh.ieeemedia@ieee.org

Joe DiNardo
Phone: +1 440 248 2456
Fax: +1 440 248 2594
Email: jd.ieeemedia@ieee.org

Midwest/Southwest (recruitment)
Darcy Giovingo
Phone: +1 847 498 4520
Fax: +1 847 498 5911
Email: dg.ieeemedia@ieee.org

Southwest (product)
Steve Loerch
Phone: +1 847 498 4520
Fax: +1 847 498 5911
Email: steve@didierandbroderick.com

Connecticut (product)
Stan Greenfield
Phone: +1 203 938 2418
Fax: +1 203 938 3211
Email: greenco@optonline.net

Southern CA (product)
Marshall Rubin
Phone: +1 818 888 2407
Fax: +1 818 888 4907
Email: mr.ieeemedia@ieee.org

Northwest/Southern CA (recruitment)
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Southeast (product)
Bill Holland
Phone: +1 770 435 6549
Fax: +1 770 435 0243
Email: hollandwfh@yahoo.com

Japan
Tim Matteson
Phone: +1 310 836 4064
Fax: +1 310 836 4067
Email: tm.ieeemedia@ieee.org

Europe (product/recruitment)
Hillary Turnbull
Phone: +44 (0) 1875 825700
Fax: +44 (0) 1875 825701
Email: impress@impressmedia.com

Advertising Personnel

Marion Delaney
IEEE Media, Advertising Director
Phone: +1 415 863 4717
Email: md.ieeemedia@ieee.org

Marian Anderson
Advertising Coordinator
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: manderson@computer.org

Sandy Brown
IEEE Computer Society,
Business Development Manager
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sb.ieeemedia@ieee.org

MICHIGAN TECH, Computer Engineering – Senior Faculty Position. The Department of Electrical and Computer Engineering at Michigan Technological University invites applications for a tenured faculty position in computer engineering. The department is seeking an established researcher in real-time computing. Areas of particular interest include real-time hardware, RTOS, and the design and implementation of embedded and/or distributed real-time systems. We are looking for a person whose central focus in real-time systems can provide technical leadership and help integrate several existing research projects involving peripheral aspects of real-time computing. A successful senior candidate will have a demonstrated track record of establishing and conducting a high quality research program, sufficient to qualify for the position of Associate or Full Professor with tenure. A candidate with a demonstrated potential for establishing a high quality research program will be considered for a position of Assistant Professor. The Department of Electrical and Computer Engineering has one of the leading undergraduate programs in the US and is aggressively growing its graduate and research programs. Michigan Tech is located in the beautiful Upper Peninsula of Michigan, offering extensive outdoor recreation. Michigan Tech is an equal opportunity employer. Send resume, statements of teaching and research interests, and contact data for three references to cpesearch@mtu.edu.

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.

UNIVERSITY OF LOUISIANA AT LAFAYETTE, The Center for Advanced Computer Studies, Faculty Positions, Graduate Fellowships. Candidates with a strong research record and an earned doctorate in computer science or computer engineering are invited to apply for multiple tenure-track assistant/associate professor faculty positions starting fall of 2008. Target areas include Grid Computing, Large Scale Data & Knowledge Engineering, Distributed Software Systems, and Entertainment Computing. Consideration will also be given to outstanding candidates in other areas. Candidates must have demonstrated potential to achieve national visibility through accomplishments in research contract and graduate students. Faculty teach mostly at the graduate and senior undergraduate levels and offer a continuing research seminar. State and university funds are available to support research initiation efforts. Salaries are competitive along with excellent support directed towards the attainment of our faculty's professional goals. The Center's colloquium series brings many world known professionals to our campus each year. The Center is primarily a graduate research unit of 17 tenure-track and 7 research faculty, with programs leading to MS/PhD degrees in computer science and computer engineering. More than 200 graduate students are enrolled in these programs. The Center has been ranked 57th in a recent NSF survey based on research and development expenditures. The Center has state-of-the-art research and instructional computing facilities, consisting of several networks of SUN workstations and other high performance computing platforms. In addition, the Center has dedicated research laboratories in Intelligent Systems, Computer Architecture and Networking, Cryptography, FPGA and Reconfigurable Computing, Internet Computing, Virtual Reality, Entertainment Computing, Software Research, VLSI and SoC, Wireless Technologies, and Distributed Embedded Computing Systems. Related university programs include the CSAB (ABET) accredited undergraduate program in Computer Science, and the ABET accredited undergraduate program in Electrical and Computer Engineering. Additional information about the Center may be obtained at http://www.cacs.louisiana.edu. A number of PhD fellowships, valued at up to $24,000 per year including tuition and most fees, are available. They provide support for up to four years of study towards the PhD in computer science or computer engineering. Eligible candidates must be U.S. citizens or must have earned an MS degree from a U.S. university. Recipients also receive preference for low-cost campus housing. Applications may be obtained and submitted at http://gradschool.louisiana.edu. The University of Louisiana at Lafayette is a Carnegie Research University with high research activity, with an enrollment of over 16,000 students. Additional information may be obtained at http://www.louisiana.edu/. The University is located

UNIVERSITY of LOUISVILLE
dare to be great

Assistant Professor in Computer Forensics & Computer Security

The Department of Computer Engineering and Computer Science invites applications for a tenure-track position at the assistant professor level to begin July 1, 2008. A Ph.D. in computer science / computer engineering, or equivalent is required with research and teaching emphasis in the areas of computer forensics, computer security, information assurance, Internet forensics, and embedded systems. Candidates are expected to have a strong teaching and research background, and will participate in interdisciplinary activities. Screening of application materials will begin 1/15/2008. Details and application information can be found at **http://louisville.edu/speed/cecs/new_web/index.shtml**

AA/EEO Employer

SUBMISSION DETAILS: Rates are $299.00 per column inch ($320 minimum). Eight lines per column inch and average five typeset words per line. Send copy at least one month prior to publication date to: Marian Anderson, Classified Advertising, *Computer* Magazine, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; (714) 821-8380; fax (714) 821-4010. *Email: manderson@computer.org.*

In order to conform to the Age Discrimination in Employment Act and to discourage age discrimination, *Computer* may reject any advertisement containing any of these phrases or similar ones: "…recent college grads…," "…1-4 years maximum experience…," "…up to 5 years experience," or "…10 years maximum experience." *Computer* reserves the right to append to any advertisement without specific notice to the advertiser. Experience ranges are suggested minimum requirements, not maximums. *Computer* assumes that since advertisers have been notified of this policy in advance, they agree that any experience requirements, whether stated as ranges or otherwise, will be construed by the reader as minimum requirements only. *Computer* encourages employers to offer salaries that are competitive, but occasionally a salary may be offered that is significantly below currently acceptable levels. In such cases the reader may wish to inquire of the employer whether extenuating circumstances apply.

in Lafayette, the hub of Acadiana, which is characterized by its Cajun music and food and joie de vivre atmosphere. The city, with its population of over 120,000, provides many recreational and cultural opportunities. Lafayette is located approximately 120 miles west of New Orleans. The search committee will review applications and continue until the positions are filled. Candidates should send a letter of intent, curriculum vitae, statement of research and teaching interests, and names, addresses and telephone numbers of at least four references. Additional materials, of the candidate's choice, may also be sent to: Dr. Magdy A. Bayoumi, Director, The Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504-4330. Tel: 337.482.6147. Fax: 337.482.5791. The University is an Affirmative Action/ Equal Opportunity Employer.

TENNESSEE TECHNOLOGICAL UNIVERSITY. The Department of Electrical and Computer Engineering (ECE) at Tennessee Technological University invites applications for a tenure-track position at the Associate or Assistant Professor level in Computer Engineering beginning in August 2008. This position includes an initial appointment as a Stonecipher Faculty Fellow in Computer Engineering. Candidates for this position must have an earned Ph.D. degree in computer engineering or closely related areas with expertise in computer engineering areas such as computer networks, parallel and distributed systems, sensor networks and/or computer security. Screening of applications will begin February 1, 2008 and continue until the position is filled. See http://www.tntech.edu/ece/jobs.html for details. AA/EEO.

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.

LOUISIANA STATE UNIVERSITY, 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

Director - School of Electrical and Computer Engineering – #07616

Located in Ithaca, N.Y., Cornell University is a bold, innovative, inclusive and dynamic teaching and research university where staff, faculty, and students alike are challenged to make an enduring contribution to the betterment of humanity.

Cornell University's College of Engineering invites nominations and applications for the position of Director of the School of Electrical and Computer Engineering. Nominations and applications of women and underrepresented minorities are especially encouraged.

Cornell University, the largest of the Ivy League institutions, has a $6B endowment, seven world class colleges, 20,000 students and world-renowned faculty. This diverse and vibrant learning community offers an extraordinary wealth of academic resources and research facilities. Located in Ithaca, NY, the picturesque campus is surrounded by the natural beauty of the Finger Lakes.

The largest of the schools and departments in the College of Engineering, the School of Electrical and Computer Engineering, is housed in Phillips Hall, Rhodes Hall, and Duffield Hall, one of the country's most sophisticated nanoscience facilities. The School benefits from a major investment in facilities, faculty, and a capital campaign with a focus on growing its significant endowment.

The School of Electrical and Computer Engineering (**www.ece.cornell.edu**) has tremendous strengths arising from its outstanding faculty, numerous Cornell-based national centers for interdisciplinary research, a balanced emphasis on science and engineering, rich traditions, and a vibrant and fostering environment that breaks down traditional department and discipline boundaries. The School graduated this past year 114 BS, 95 MEng, 20 MS, and 29 PhD students.

The Director is the chief academic and administrative officer of the School providing professional leadership and example with responsibility for overseeing the school governance; curriculum and program development; faculty and student recruitment, development and retention; financial and facilities management, and external relations to enhance the School's reputation and position.

The ideal candidate will possess the following characteristics:

- An earned doctorate degree in electrical or computer engineering or a related field
- A record of excellence as a scholar including outstanding research and a strong commitment to teaching
- The ability to engage faculty to develop a common vision and to successfully influence change to achieve that vision
- Demonstrated leadership and administrative effectiveness including experience in strategic planning, goal setting, fiscal management, and the attainment of organizational objectives
- Excellent communication, problem-solving, conflict management, and negotiation skills with a record of building effective relationships with faculty, staff, alumni, and industry leaders
- Demonstrated commitment to fostering an environment that supports equity and diversity
- The ability to contribute to fund raising activities by inspiring interest in, and commitment to, the School

Applications should include a cover letter, a statement that describes the candidate's interest and qualifications for the position and a curriculum vita. The position will remain open until filled. Review of materials will begin immediately and continue until the new director is selected. Nominations and applications should be sent electronically to **ecedirectorsearch@cornell.edu**

Cornell University is an Affirmative Action/ Equal Opportunity Employer and Educator.

http://chronicle.com/jobs/profiles/2377.htm

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, 2008 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. Iyengar, Co-Chair of Center for CyberSpace Security, Department of Computer Science, 298 Coates Hall, Louisiana State University, Ref: #026921 & #023602, Baton Rouge, LA 70803. E-mail: search1@csc.lsu.edu. LSU IS AN EQUAL OPPORTUNITY/EQUAL ACCESS EMPLOYER.

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 contribution 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.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Colorado Springs, CO. Systems Software Engineer V. Reqs. Windows, Linux, UNIX op sys, Qualification/Test Exp of SW, Firmware, and/or Hardware, Test Automation, CMM, Storage Area Networks (SAN), Fibre Channel, SCSI Protocol, LAN/WAN. Reqs. incl. Bachelor's degree or foreign equiv. in CS, CE or related & 6 years of related exp. Send resume & refer to job #COLJDA. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

SOFTWARE ENGINEERS, PROGRAMMER ANALYSTS. Analyze, dsgn, dvlp & support software applics using J2EE, Websphere App. Server, Apache/Tomcat, JSP/Servlets, Struts, JDBC, JMS, EJBs, XML, WSAD, PL/SQL. $50.68/hr. Bach deg in Engg or equiv. Min 5 yrs exp in skills to perform reqd job duties. Resume to Serebrum Corp, 555 Rt1 South, Iselin, NJ 08830.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Cupertino, CA. Senior QA Engineer. Reqs. SW dvlpmt, Manual and Automa-

tion processes and tools, Build Verification, Oracle database, Configuration of machines w/various Op sys and java. Reqs. incl. Bachelor's degree or foreign equiv. in CS, MIS or related & 5 years of related exp. Send resume & refer to job #CUPVNA. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Cupertino, CA. Service/Support Off-Site Engineer III. Reqs. HTTP Technologies, UNIX admin, C, C++ programming, Weblogic, Websphere, MSSQL, Oracle, IIS, Tomcat, Sun Solaris, HP/UX and Red Hat Linux. Reqs. incl. Master's or foreign degree equivalent in CS, CE, EE or a related field. Send resume & refer to job #CUPMBA. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Cupertino, CA. Software Engineering Lead/Manager-Platform. Reqs. Java, J2EE and Infrastructure dvlpmt. Reqs. incl. Bachelor's or foreign degree equiv in CS, EE, or a related field and 8 years of related exp. Send resume & refer to job #CUPAGR. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Palo Alto, CA and various unanticipated sites throughout the U.S. Technology Consultant. Reqs. Java, J2EE, SQL, Infrastructure dvlpmt, UNIX and Windows. Reqs. incl. Master's degree or foreign equiv in CS, CE or related, and 3 yrs of related exp, or a Bachelor's degree or foreign equiv and 5 yrs of related exp. Send resume & refer to job #PALJKA. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Palo Alto, CA and various unanticipated sites throughout the U.S. Business Value Consultant. Reqs. IT Mgmt, ITIL, Cobit, CMm, Six Sigma, ROI, NPV and IRR. Reqs. incl. Master's or foreign degree equiv in Busi Admn, Finance or a related field and 5 years related exp. Send resume & refer to job #PALPBE. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

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.

SENIOR SOFTWARE ENGINEER F/T sought by Music Entertainment Co. (Palo Alto, CA) Supv & dvlp s/ware solutions for music catalog & commerce platform. Bach deg in Comp Sci or Engg (or for. equiv. deg) +5 yrs post-baccalaureate progressive exp. in job offd is reqd. La La Media, Inc., Attn. William Alvarado, CFO, 209 Hamilton Ave., Palo Alto, CA 94301. No calls.

ADVANCED MICRO DEVICES, INC. is accepting resumes for the following: In Sunnyvale & Santa Clara, CA: Design Eng

(CA28008), Sr. Design Eng (CA28003), Principal Design Eng (CA28001), MTS, Design Eng (CA28014), Systems Eng (CA28006), Process Development Eng (CA28024), Sr. Process Development Eng (CA28009), MTS, Process Development Eng (CA28020), Sr. Semiconductor Packaging Eng (CA28018), Manufacturing Eng (CA28025), Application Engineer (CA28026), Lead Source Mgr. (CA28027), Technology & Integration Eng. (CA28028), Sr. Technology & Integration Eng (CA28015), MTS, Technology & Integration Eng (CA28016), Sr. Reliability Eng. (CA28029), MTS, SW Eng. (CA28030). In Fort Collins, CO: Design Eng. (FC28001). In Austin, TX: Senior Program Manager (T28613). Send resume with job title and code reference to: AMD, Application Mail Stop 101, One AMD Place, P.O. Box 3453, Sunnyvale, CA 94088.

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM, Department of Computer and Information Sciences, Assistant/Associate Professor. The Department of Computer & Information Sciences at the University of Alabama at Birmingham (UAB) is seeking candidates for a tenure-track/tenure-earning faculty position at the Assistant or Associate Professor level beginning August 15, 2008. Candidates with expertise in Artificial Intelligence who could interact with existing research groups in Knowledge Discovery and Data Mining, Computer Graphics and Imaging, and Software Engineering are of particular interest. Also potential for multidisciplinary collaboration with research groups working in Bioinformatics and Computer Forensics would be advantageous. For additional information about the department please visit http://www.cis.uab.edu. Applicants should have demonstrated the potential to excel in one of these areas and in teaching at all levels of instruction. They should also be committed to professional service including departmental service. A Ph.D. in Computer Science or closely related field is required. Applications should include a complete curriculum vita with a publication list, a statement of future research plans, a statement on teaching experience and philosophy, and minimally four letters of reference with at least one letter addressing teaching experience and ability. Applications and all other materials may be submitted via email to facapp@cis.uab.edu or via regular mail to: Search Committee, Department of Computer and Information Sciences, 115A Campbell Hall, 1300 University Blvd., Birmingham, AL 35294-1170. Interviewing for the position will begin as soon as qualified candidates are identified, and will continue until the position is filled. The department and university are committed to building a culturally diverse workforce and strongly encourage applications from women and individuals from underrepresented groups. UAB has an active NSF-supported ADVANCE program and a Spouse Relocation Program to assist in the needs of dual career couples. UAB is an Affirmative Action/Equal Employment Opportunity employer.

SOFTWARE DEVELOPER sought by S/ware Consulting (Santa Clara, CA) Req BS in EE or its for. equiv. +5 yrs exp in s/ware applic dvlpmt in corp, transaction based systms envrmt. Knowl of applied core & J2EE dsgn patterns, distributed architectures; J2EE technologies, incl EJB, Servlets; Service Oriented Architecture knowl, Applic Server, Ajax prototype, DWR, open source tech solutions incl Struts, Spring, Hibernate; Build & deployment strategies incl Ant, Maven. Apply to Integnology Corporation, Jamil Ahmed, COO, 3945 Freedom Circle, Ste 400, Santa Clara, CA 95054.

TEXAS A&M UNIVERSITY, Department of Computer Science, Tenure/Tenure-Track Faculty Positions. Texas A&M University is at the end of a five-year growth campaign to hire 447 faculty members as part of its historic Vision 2020 plan to establish the University as a consensus top-ten public institution. This campaign includes over 100 new positions for the Dwight Look College of Engineering. As part of the expansion, the Department of Computer Science is recruiting for multiple tenure-track positions at all levels: assistant, associate, and full professor. Exceptional candidates will be considered in all areas, but special consideration will be given to those in Security. We are also looking for distinguished candidates at the level of full professor. Qualified candidates for all positions must have a Ph.D., and will be expected to teach courses, mentor graduate students, and establish a vibrant research program with substantial impact and external funding. The Department of Computer Science has 38 tenure-track faculty members currently. The faculty holds over 60 important and influential professional positions, including editorships for scientific journals and general chairs of technical conferences. The faculty is also well-recognized for contributions to their fields, with research known throughout the international academic community and global industry alike. The department currently has one National Academy of Engineering member, five IEEE fellows, one ACM Fellow, and ten PYI/NYI/CARREER awardees. Texas A&M University is centrally located in College Station, Texas, which is roughly equidistant from three of the 10 largest cities in the United States (Houston, Dallas and San Antonio) as well as the State Capital (Austin). Texas A&M ranks in the top-20 U.S. institutions for the enrollment of National Merit Scholars. Enrollment includes approximately 45,000 students, with 8,700 pursuing graduate degrees. Each year, Texas A&M's 2,500 faculty conduct more than $500 million worth of sponsored research projects. Additional information about faculty recruiting is available at http://www.cs.tamu.edu/facprospective. Prospective candidates should apply online at https://apply2.cs.tamu.edu/gts/applicant/faculty/. For questions about the positions, please contact: search@cs.tamu.edu. Texas A&M is an equal opportunity/affirmative action employer and actively seeks candidacy of women and minorities. Applications are also encouraged from dual-career couples.

UNIVERSITY OF CALGARY, 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 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, Canada, T2N 1N4 or via email to: search@cpsc.ucalgary.ca. Applications will be reviewed immediately and will continue 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 http://www.ucalgary.ca/hr/careers.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Bellevue, WA. Pre-Sales Consultant IV. Reqs. ITIL, Application Mgmt., Network

topologies and schematics. Reqs. incl. Bachelor's degree or foreign equiv. in BA, Communications, Info Sys or related & 5 years of related exp. Send resume & refer to job #BELLKI. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

THE OHIO STATE UNIVERSITY, Department of Computer Science and Engineering, Assistant Professor. 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.

JDS UNIPHASE has the following job opportunities available (various levels/types) in San Jose, CA (1); Milpitas, CA (2); Atlanta, GA (3); Germantown, MD (4); and Renton, WA (5): Business Analyst (BA1), (BA2); IT Professional (ITP1), (ITP2); Principal Engineer (PE1), (PE2); Sr. Engineers (SRE1), (SRE2); Software Development/Research Engineer (SWD/RE3); Technical Instructor (TI4); Engineer (E4); E2 Software Engineer (ESWE4); Software Engineer (SE4); QA Engineers (TE5). Positions may require travel. Submit resume to 430 N. McCarthy Blvd., Milpitas, CA 95035, Attn: VJ/2.1.1067. Must reference job title and job code (ie SE4) in order to be considered. EOE. www.jdsu.com.

HEWLETT – PACKARD COMPANY is accepting resumes for the following positions in Palo Alto, CA: Project Manager (Reference #PALMRA) and Business Analyst (Reference #PALBGO). In Cupertino, CA: Information Technology Analyst (Reference #CUPYLI), Business Systems Analyst (Reference #CUPOLU). Please send resumes with reference number to Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

SOFTWARE ENGINEER F/T in Poughkeepsie, NY. Must have Master's deg or equiv in Information Technology & 1 yr of exp developing applications using J2EE, Servlets, Jsp, JClass, Web Sphere5.1, WSAD 5.1.1, DB2, UNIX, PVCS Tracker, PVCS Version Manager, Exceed 10.0, Application Expert. Send resume: Apollo Consulting Services Corp., Recruiting (JP), 14 Catharine St, Poughkeepsie, NY 12601.

MISSISSIPPI STATE UNIVERSITY, Head, Department of Computer Science and Engineering. Applications and nominations are being sought for the Head of the Department of Computer Science and Engineering (www.cse.msstate.edu) at Mississippi State University. This is a 12-month tenure-track position. Part of the Bagley College of Engineering, the department has approximately 325 undergraduate majors, 70 graduate students, and 18 tenured and tenure-track faculty. The department offers undergraduate programs in Computer Science and Software Engineering, and jointly administers the undergraduate program in Computer Engineering with the Department of Electrical and Computer Engineering. At the graduate level, we offer M.S. and Ph.D. degrees in Computer Science and faculty also direct graduate students in Computational Engineering and Computer Engineering. Certificates in Software Engineering, Information Assurance, and Computational Biology are also available. Research expenditures total about $3 million dollars annually and the university as a whole is ranked 48th among 271 U.S. institutions in computer science expenditures. Research areas for the department are high-performance computing, artificial intelligence, graphics and visualization, computer security, and software engineering. Three current faculty members have received NSF CAREER awards. Our computer security area has been designated a National Center of Academic Excellence in Information Assurance Education (CAEIAE) by the National Security Agency (NSA). Mississippi State University is a comprehensive land-grant institution with approximately 17,000 students and about 1,000 faculty members. The university is a leader in high performance computing, housing a supercomputer in the top 20 among U.S. universities. The university's main campus is

NATIONAL SUN YAT-SEN UNIVERSITY, TAIWAN

Faculty Opening at Department of Computer Science and Engineering

The Department of Computer Science and Engineering at National Sun Yat-sen University invites applications for tenure-track positions from August 2008. Applicants in all areas of computer science and engineering are sought.

Applicants for assistant professorship must demonstrate strong research potential, in addition to good teaching ability. Applicants for associate professorship and professorship must have an exceptional record of research achievement. All successful candidates are expected to conduct research projects and to teach for the department as well. The department offers BS, MS and Ph. D. degrees in Computer Science and Engineering. The official language of teaching is Chinese, and English teaching is encouraged by the school. For more information, please visit our web site: **http://www.cse.nsysu.edu.tw**.

Applications should include a curriculum vitae, recent publications, and reference letters from at least three people who can comment on the applicant's professional qualification. Applications for assistant professorship should also include academic transcripts. Applications should be sent to:

Professor Chun-Hung Richard Lin, Chairman
Department of Computer Science and Engineering
National Sun Yat-sen University
Kaohsiung, Taiwan 80424

E-mail: lin@cse.nsysu.edu.tw
TEL: +886-7-5252000 ext. 4339
FAX: +886-7-5254301

The deadline for applications is January 20, 2008.

located in Starkville, Mississippi, a vibrant community approximately 2 hours from Jackson MS, Birmingham AL, and Memphis TN. The successful Head will provide: •Vision and leadership for nationally recognized computing education and research programs, •Exceptional academic and administrative skills, •A strong commitment to faculty recruitment and development. Applicants must have a Ph.D. in computer science, software engineering, computer engineering, or a closely related field. The successful candidate must have earned national recognition by a distinguished record of accomplishments in computer-science education and research. Demonstrated administrative experience is desired, as is teaching experience at both the undergraduate and graduate levels. The successful candidate must qualify for the rank of professor. Please provide a letter of application outlining your experience and vision for this position, a curriculum vita, and names and contact information of at least three professional references. Application materials should be submitted online at http://www.jobs.msstate.edu/. Screening of candidates will begin February 22, 2008 and will continue until the position is filled. Mississippi State University is an AA/EOE institution. Qualified minorities, women, and people with disabilities are encouraged to apply.

NETWORK ENGINEER for administrative and technical services company in Glendale, CA. Apply to jobpost@mail.all-in-1.com, Job#0712-01.

HOFSTRA UNIVERSITY. The Department of Computer Science at Hofstra University (www.cs.hofstra.edu) invites applications for one or more tenure track faculty positions at the Associate/Assistant professor level. Strong candidates in all areas of computer science will be considered. Junior applicants are expected to have completed a Ph.D. in computer science (or a related field) by the start of their contract. Senior applicants should be willing to serve as director of a new distance learning MS program. The department offers BA, BS, MA and MS programs in computer science and a BS in computer engineering. Hofstra University is a liberal arts institution located on Long Island, approximately 25 miles from Manhattan. Please arrange three letters of recommendation to be sent electronically and in hard copy to the address provided. Send electronically the application consisting of a cover letter, a curriculum vitae, and statements of research interests and teaching. Dr. Gretchen Ostheimer, Chair, Faculty Search Committee, Department of Computer Science, 103 Hofstra University, Hempstead, New York 11549-1030 USA, cscsearch@hofstra.edu. Hofstra University is an equal opportunity employer, committed to fostering diversity in its faculty, administrative staff and student body, and encourages applications from the entire spectrum of a diverse community.

TIBCO DEVELOPER. Design, and develop software in support of health care insurance applications using Tibco; program, test, implement, modify & maintain computer system components (e.g. databases, extract files, programs) in support of software applications; ensure applications or database services are readily available to users across multi-platform distributed system or mainframe environment. Requires M.S. in Comp. Sci. or rel'd tech field plus 2 yrs post baccalaureate exp as a Software Engineer including at least 1 yr using Tibco w some exp in support of health care applications or related insurance services. Mail resume to: Great-West, 8525 E. Orchard Rd., Attn: R. Hanna (IEEE)-1T3, Greenwood Village, CO 80111. EOE.

NXP SEMICONDUCTORS has the following job opportunities available (various levels/types). SAN JOSE, CA: Application Engineers (AE-CA), Design Engineers (DE-CA), Software Engineers (SWE-CA). Austin, TX: Applications Engineer (AE-TX), Design Engineer (DE-TX). Middle Island, NY: Field Application Engineer (FAE-NY). Hopewell Junction, NY: Yield Engineer (YE-NY). Some positions may require travel. Submit resume by mail to PO Box 4115, Santa Clara, CA 95056-4115, Attn: HR Coordinator. Must reference job title and job code (i.e. SWE-CA) in order to be considered. EOE.

THE UNIVERSITY OF ALABAMA, Department of Computer Science. The University of Alabama, Department of Computer Science, invites applications for a new assistant professor position to begin August 16, 2008. Candidates must have an earned Ph.D. in computer science or a related field, with solid evidence of superior research and scholarship accomplishments that are appropriate for the desired level of appointment, as well as quality teaching abilities. Applicants from all areas of computer science will be considered. Those who specialize in software engineering, database systems, operating systems, or networking are particularly encouraged to apply. The University of Alabama, located in Tuscaloosa, is considered the Capstone of higher education and is also the largest institution in the State. The Department of Computer Science, housed in the College of Engineering, currently has twenty-three faculty members (15 tenured/tenure-track), roughly 200 undergraduates in an ABET accredited B.S. degree program, and approximately 50 M.S. and Ph.D. students. Outstanding applicants should send curriculum vitae and the names and addresses of at least three references to the address shown below. E-mail submissions are also encouraged. Faculty Search Committee, Department of Computer Science, Box 870290, The University of Alabama, Tuscaloosa, AL 35487-0290. E-mail: faculty.search@cs.ua.edu. For additional information, please visit http://cs.ua.edu or contact the Search Committee at faculty.search@cs.ua.edu. Review of applications will begin January 25, 2008 and will continue until the position is filled. The University of Alabama is an equal opportunity/affirmative action employer. Women and minorities are particularly encouraged to apply.

HEWLETT - PACKARD COMPANY has an opportunity for the following position in Houston, Texas. Business Planning Manager. Reqs. project mgt exp; business S/W dvlpmt exp; strong financial & acctg skills; & business consulting exp. Reqs. incl. Bachelor's degree (preferred) or foreign equiv. in Business Admin., Public Admin., MIS, or related field of study & 2-6 years of related exp. Send resume & refer to job #HOURVE. Please send resumes with job number to Hewlett-Packard Company, 19483 Pruneridge Ave., MS 4206, Cupertino, CA 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

ACCOUNT & PAYMENT REPRESENTATIVE. Description: As part of our expansion program a small company is looking for Account & Payment representative, it pays $3000 a month plus benefits and takes only little of your time. Please contact us for more details. Requirements - Should be a computer Literate. 2-3 hours access to the internet weekly. Must be over 19yrs of age. Must be Efficient and Dedicated. If you are interested and need more information, Contact (P R O)Stanley, Email: stevematte00@yahoo.com.

MANAGER, TECHNOLOGY, Culver City, CA: Analyze, design, develop, manage product development across platforms using Java, Java based tools. Work with development team, allocate resources, maintain project schedules. Work with content management systems in multiplatform environment. Gather, respond to user requirements. Troubleshoot, test, maintain products/systems. Reply to: Edgesoft, Inc., 6133 Bristol Parkway, suite 301, Culver City, CA 90230.

HEWLETT – PACKARD COMPANY is accepting resumes for the following position in Houston, TX: Business Systems Analyst (Reference #HOURGA). Please send resume with reference number to

Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT – PACKARD COMPANY is accepting resumes for the following position in Alpharetta, GA: Business Systems Analyst (Reference #ALPSKU). Please send resume with reference number to Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT – PACKARD COMPANY is accepting resumes for the following position in Roseville, CA: IT Developer Engineer (Reference #ROSPRA). Please send resume with reference number to Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT – PACKARD COMPANY is accepting resumes for the following position in San Diego, CA: Hardware Development Engineer (Reference#SDYTO). Please send resume with reference number to Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

SENIOR SYSTEMS ADMINISTRATOR F/T (Poughkeepsie, NY). Must have Bach deg in Comp Sci, Elec. Engg or related & 1 yr exp using RS/6000, AIX, and Windows NT. Send resume: Apollo Consulting Services Corp., Recruiting (AS), 14 Catharine St, Poughkeepsie, NY 12601.

MISSISSIPPI STATE UNIVERSITY, FACULTY POSITION IN COMPUTATIONAL BIOLOGY. The Department of Computer Science and Engineering (http://www.cse.msstate.edu) invites applications for a tenure-track faculty position at the Assistant or Associate Professor level in the area of computational biology. Candidates for the position are expected to hold a Ph.D. in computer science, computational biology, or a closely related field (ABDs may be considered). The person filling this position will be affiliated with the MSU Institute for Digital Biology (IDB), a university-level multi-disciplinary research institute that merges MSU's strengths in engineering and biology to solve problems related to health, nutrition, biofuels, food safety, bio-security and agriculture. IDB faculty are currently funded by NSF, USDA, DOD, NIH, and DOE. MSU has a state-of-the art facility for genomics and proteomics (the Life Science and Biotechnology Institute). Research areas within the department include high performance computing, artificial intelligence, graphics and visualization, computer security, and software engineering. Mississippi State University is the largest university in the State of Mississippi with approximately 1000 faculty and 17,000 students. The Department of Computer Science and Engineering has 18 tenured and tenure-track faculty and offers academic programs leading to the B.S., M.S. and Ph.D. in Computer Science, and a B.S. in Software Engineering. The department offers a certificate in Computational Biology and jointly administers the B.S. in Computer Engineering. Faculty also direct Ph.D. students in Computational Engineering and Computer Engineering and work with a number of additional on-campus research centers including the High Performance Computing Collaboratory, the Institute for Neurocognitive Science and Technology, and the Sustainable Energy Center. Seven faculty members in the department have been recognized by NSF CAREER awards. Department research expenditures total around three million dollars per year. Mississippi State University is ranked 48th among 271 U.S. institutions in R&D expenditures in engineering by the National Science Foundation. Please provide a letter of application, a curriculum vita, research and teaching statements, and names and contact information of at least three professional references. Application materials should be submitted online at http://www.jobs.msstate.edu/, or emailed to office@cse.msstate.edu . Screening of candidates will begin February 28, 2008 and will continue until the position is filled. Mississippi State University is an AA/EOE institution. Qualified minorities, women, and people with disabilities are encouraged to apply.

GEORGETOWN UNIVERSITY, Senior Faculty Position and Chair of Department, 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. For more information, please visit our website at http://www.cs.georgetown.edu/.

HEWLETT – PACKARD COMPANY is accepting resumes for the following position in Houston, TX: Technical Analyst (Development Lead) (Reference # HOUNMA). Please send resumes with reference number to Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

HEWLETT – PACKARD COMPANY is accepting resumes for the following position in Alpharetta, GA: Technical Analyst (Reference # ALPSAR). Please send resumes with reference number to Hewlett-Packard Company, 19483 Pruneridge Avenue, Mail Stop 4206, Cupertino, California 95014. No phone calls please. Must be legally authorized to work in the U.S. without sponsorship. EOE.

SOFTWARE ENGINEER, Applications for CRM applications. Job site Glendale, CA. Send resume to Attn: C.Ng, MIS- PO Box 29048, Glendale, CA 91209-9048.

Computer Society Announces 2008 Program Slate

The IEEE Computer Society has been at the forefront of major developments in computing since 1946. With more than 40 percent of its members living and working outside the US, the Computer Society encourages and facilitates international cooperation, communication, and information exchange. In 2008, the Society again offers a full catalog of periodicals that address all aspects of computer science and engineering, including 14 magazines, 17 transactions, and one letters publication. The Society also publishes cutting-edge papers and articles from more than 200 conferences, covering topics in all areas of computer science and engineering. All members receive a subscription to *Computer* magazine as a benefit of membership.

VOLUNTEERS LEAD PUBLICATIONS

Members can become more involved with the Society's publications program by serving as authors, reviewers, or editorial board members for individual magazines and journals. Editorial boards work together with professional staff to provide members with the targeted, high-quality content that has been the hallmark of Computer Society publications for more than 60 years.

Several magazines are seeking new editors in chief for 2009. To find out more about these positions, see "Society Publications Seek Editors in Chief for 2009-2010 Terms." Visit the Computer Society web site at www.computer.org/publications to learn more about the periodicals of your choice.

COMPUTER SOCIETY ONLINE DIGITAL LIBRARY

The IEEE Computer Society Digital Library provides online access to 25 current and former Society magazines, transactions, and letters, more than 2,500 selected conference proceedings, and many tutorials and scholarly books. All members enjoy free online access to *Computer*. An all-in-one CSDL account, available in 2008 for $121, offers members access to content from any Computer Society publication. This combined resource is available to IEEE Computer Society members and library/institution customers only. Traditional single-magazine subscriptions are also available in print, online, or combined formats. Individual documents are available for $19.

Access to the full text of individual IEEE Computer Society periodicals is available to members who subscribe and have a valid IEEE web account.

Some IEEE Computer Society periodicals are published in technical cosponsorship with other IEEE societies, and CSDL search results now provide links to materials from the ACM. Article abstract pages include links to resources listed in the bibliographies, and CSDL subscribers can save searches and search terms, making it easier to find new material on a given topic as it appears. Visit www.computer.org/csdl for further details.

Nominations Open for IEEE Division V Director-Elect

IEEE Computer Society members are invited to submit nominations for candidates to serve as 2009 IEEE Division V director-elect and 2010-2011 Division V director.

Division directors represent the members of IEEE societies on the IEEE Board of Directors and the Technical Activities Board; Division V and VIII directors represent the Computer Society membership. Elections for Division V director are typically held on even-numbered years, and Division VIII elections are held on odd-numbered years. The elected representative then serves one year in the director-elect role before assuming a two-year division director term.

Past Computer Society president Deborah Cooper currently serves as IEEE Division V director for 2008-2009. Past Computer Society president Thomas Williams, of Synopsys, currently serves as IEEE Division VIII director for 2007-2008. In a recent vote, IEEE members chose another former Computer Society president, Stephen Diamond, as Division VIII director-elect for 2008.

Submit nominations by **11 January** to Michael Williams, Chair, Nominations Committee, IEEE Computer Society, 1730 Massachusetts Ave. NW, Washington, DC 20036-1992 or nominations@computer.org.

E-LEARNING CAMPUS

The IEEE Computer Society e-Learning Campus is another no-cost benefit of membership. Computer Society members have free access to more than 1,300 web-based courses that range from primers on business and office fundamentals, to surveys of security strategies, to specialized technical courses for computing professionals. Offered in partnership with Thomson NETg, courses in the e-Learning catalog offer presentations with voiceovers, 3D graphics, flash animations, onscreen text, and visual sentences that turn complex concepts into easy-to-understand images. Users can take preassessment tests, track ongoing progress, and study at a self-directed pace.

Members can also access books, certification courses, and other study materials. Twice each year, volunteers review and select new e-Learning Campus offerings based on survey data and usage numbers.

Books 24x7 collections

Members of the Computer Society enjoy access to a database of more than 500 technical books via a partnership with Books 24x7. Titles cover a broad spectrum of topics in computing.

Members of IEEE or the IEEE Computer Society can now purchase a 12-month subscription to two specialized libraries containing more than 6,200 unabridged books on a variety of technical and engineering topics, including books from IEEE Press. These two libraries, the ITPro (5,500 titles) and EngineeringPro (700 titles) collections, are offered exclusively to IEEE members at a special discounted rate.

To learn more about IEEE Computer Society e-learning opportunities, visit the online campus at www.computer.org/distancelearning.

Certifications

Intended for midlevel software development and software engineering professionals, the IEEE Computer Society Certified Software Development Professional program offers the sole brand-name professional credential in software development. The program encompasses exam-based testing to demonstrate mastery of a well-defined body of knowledge, as well as verification of both a solid experience base in the field and recent continuing professional education work. Visit www.computer.org/certification for complete program information, including fees and test dates.

Online exams available via BrainBench offer practical certifications in areas that include office productivity, business fundamentals, and Cisco networks.

IEEE ReadyNotes

IEEE ReadyNotes are PDF-based guidebooks and tutorials that serve as a quick-start reference for busy computing professionals. Sample titles include *Designing and Implementing Softcoded Values*, *Evaluating the Performance of Software Engineering Professionals*, and *Introduction to Python for Artificial Intelligence*. IEEE ReadyNotes sell for $19 or less. Find the ReadyNotes catalog at www.computer.org/readynotes.

CONFERENCES

IEEE Computer Society technical councils, task forces, and technical committees sponsor the majority of the Society's technical meetings. The following selection of high-profile conferences is a cross-section of the many events presented by the society each year.

8-12 March: IEEE VR 2008

VR2008

IEEE VR 2008 is the leading international conference and exhibition in virtual reality. It provides a unique opportunity for interaction among leading experts in virtual reality and closely related fields such as augmented reality, mixed reality, and 3D user interfaces.

Located this year in Reno, Nevada, VR 2008 will take place in conjunction with the Symposium on Haptic Interfaces and the IEEE Symposium on 3D User Interfaces. Topics at conference-related workshops include virtual cityscapes, massively multiuser virtual environments, and software engineering for interactive systems.

VR 2008 leads the Computer Society Technical Committee on Visualization and Graphics' annual calendar of conferences and workshops. Advance registration fees are $870 for members, $1,120 for nonmembers, and $350 for student members. For further details, visit http://conferences.computer.org/vr/2008.

31 March- 4 April: ECBS 2008

ECBS 2008
Making systems more effective...

The 15th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems is devoted to the latest research findings in the design, development, deployment, and analysis of complex systems that are largely controlled by computers.

Keynote speakers at ECBS 2008 include Noel Sharkey of the University of Sheffield, Brian Randell of the University of Newcastle upon Tyne, and John Strassner of Motorola Research Labs.

ECBS is colocated with the 13th IEEE International Conference on the Engineering of Complex Computer Systems and the 5th IEEE International Workshop on the Engineering of Autonomic and Autonomous Systems.

Sponsored by the IEEE Computer Society Technical Committee on the Engineering of Computer-Based Systems, the 2008 conference will take place in Belfast, Northern Ireland. Visit www.compeng.ulster.ac.uk/events/ecbs2008 for more information.

7-12 April: ICDE 2008

The 24th IEEE International Conference on Data Engineering continues as a premier forum for presenting research results on advanced data-intensive applications and discussing issues in data and knowledge engineering. Conference participants share research solutions and cooperate to identify new issues and directions for future research and development work.

Organizers have called for submissions on topics that include ubiquitous data management, mobile databases, query processing and optimization, data structures and data management algorithms, and XML data processing, filtering, routing, and algorithms.

The IEEE Computer Society Technical Committee on Data Engineering sponsors ICDE 2008 in conjunction with corporate sponsors Microsoft and CIC. Regular conference fees, including workshops, total $700 for members, $875 for nonmembers, and $375 for student members. Visit the ICDE 2008 web site at www.icde2008.org for further details.

14-18 April: IPDPS 2008

The 22nd International Parallel and Distributed Processing Symposium is a forum where researchers, scientists, and engineers present and discuss new findings in parallel processing and distributed computing. The five-day conference features contributed papers, panels, invited speakers, and commercial presentations.

IPDPS participants will also have the opportunity to organize informal birds-of-a-feather sessions. Scheduled workshop topics at IPDPS 2008 include high-level parallel programming models, nature-inspired distributed computing, and communication architecture for clusters. New for 2008 is the PhD Forum, introduced by the IEEE Computer Society Technical Committee on Parallel Processing as an opportunity for graduate students to present their proposed and/or partially completed dissertation work to a broad audience of both academic and industrial researchers and practitioners.

IPDPS 2008, located this year in Miami, is sponsored by the TCPP in cooperation with ACM Sigarch. To learn more about IPDPS 2008, visit www.ipdps.org.

16-18 April: Cool Chips XI

The 2008 IEEE Symposium on Low-Power and High-Speed Chips is a high-profile international symposium for presenting recent advancements in all areas of microprocessors and their applications.

This year, the Cool Chips symposium will focus on the architecture, design, and implementation of chips in areas including multimedia, digital consumer electronics, mobile, graphics, encryption, robotics, networking, and biometrics. Cool Chips XI organizers have also solicited original works on novel software solutions to challenges that include binary translations, compiler issues, and low-power techniques.

Cool Chips XI, which takes place in Yokohama, Japan, is sponsored by the IEEE Computer Society Technical Committees on Microprocessors and Microcomputers, and Computer Architecture. Cosponsors include the Institute of Electronics, Information, and Communication Engineers Society, ACM Sigarch, and the Information Processing Society of Japan. For more conference information, visit www.coolchips.org.

10-18 May: ICSE 2008

The 30th International Conference on Software Engineering offers leading researchers, practitioners, and educators an opportunity to present and discuss recent trends in software engineering.

Opportunities for professional engagement include workshops, tutorials, research demonstrations, exhibits, paper tracks on research and education, and special tracks on telecommunications, automotive, and healthcare systems. Also scheduled are a doctoral symposium and new faculty symposium.

Several other software engineering-related conferences and symposia are scheduled to run concurrently with ICSE, including the 2008 International Conference on Software Process. The IEEE Computer Society Technical Council on Software Engineering sponsors ICSE in cooperation with ACM Sigsoft. Visit http://icse08.upb.de for more conference information.

18-21 May: Security and Privacy 2008

In its 28th year, the IEEE Symposium on Security and Privacy is a premier forum for bringing together researchers and practitioners to present cutting-edge developments in computer security and electronic privacy.

For 2008, conference organizers have solicited papers on topics that include cryptographic protocols, database security, user authentication, and application-level security, among others. Papers presented at the conference represent advances in the theory, design, implementation, analysis, and empirical evaluation of secure systems, both for general use and for specific application domains.

The Security and Privacy conference, which takes place in Berkeley, California, is sponsored by the IEEE Computer Society Technical Committee on Security and Privacy, in cooperation with the International

Association for Cryptologic Research. See www.ieee-security.org/TC/SP2008/oakland08.html for complete conference details.

19-22 May: CCGrid 2008

CCGrid

The Eighth IEEE International Symposium on Cluster Computing and the Grid, the latest in a series of successful international conferences that began in 2000, provides researchers and practitioners with an opportunity to share their research and experiences in overcoming challenges in web services and grid technology.

Several workshops and other events complement the larger conference. Workshop topics include high-performance grid networks, computer tools for bioscience, networks and distributed computing platforms, and autonomics for grids and data centers. Taking place in conjunction with CCGrid 2008, the IEEE Computer Society Technical Committee on Scalable Computing Doctoral Symposium provides a forum for students in scalable computing to obtain feedback on their dissertation topics and advice on initiating a research career.

CCGrid 2008, taking place this year in Lyon, France, is sponsored by the IEEE Computer Society TCSC. Visit www.ens-lyon.fr/LIP/RESO/ccgrid2008 for program highlights and more conference information.

17-20 July: ICDCS 2008

The International Conference on Distributed Computing Systems, the first conference series in the field of distributed computing systems, has provided a forum for engineers and scientists in academia, industry, and government to discuss the latest research findings on topics including agents and mobile code, middleware, and ubiquitous computing. ICDCS 2008 will take place in Beijing.

Conference organizers have solicited papers on topics that include cyberinfrastructure for distributed computing, sensor networks and applications, wireless and mobile computing, multimedia systems, and distributed cyber/physical systems. In its 28th year, ICDCS is presented by the IEEE Computer Society Technical Committee on Distributed Processing, in cooperation with the Chinese Academy of Sciences' Institute of Computing Technologies. Visit www.engin.umd.umich.edu/icdcs for registration and detailed program information as it becomes available.

28 July–1 August: SAINT 2008

SAINT

The 2008 Symposium on Applications and the Internet will draw researchers from around the world to share new ideas and findings regarding the Internet and its applications. Participants come from a spectrum of disciplines in government, industry, and academia.

SAINT organizers have called for papers on topics that include content management, ubiquitous and pervasive computing, Internet communities, network and protocol architectures, and information appliances. SAINT 2008—taking place this year in Turku, Finland—will share its venue with COMPSAC 2008, the IEEE Computer Society Conference on Computer Software and Applications.

Cosponsored by the Information Processing Society of Japan, SAINT is the flagship conference of the IEEE Computer Society Technical Committee on the Internet. Visit www.icta.ufl.edu/saint08 for more conference information.

Proceedings from many conferences are available through the Computer Society Digital Library. CSDL subscribers enjoy full access to an online collection that also includes all Computer Society magazines and most Computer Society journals. Nonsubscribers can search the collection at www.computer.org/csdl and purchase individual documents. IEEE Computer Society members also enjoy as much as a 25 percent discount on registration fees at Society-sponsored conferences. Visit www.computer.org for complete details on all IEEE Computer Society publications, conferences, symposia, technical meetings, volunteer opportunities, and other activities.

Society Publications Seek Editors in Chief for 2009-2010 Terms

The IEEE Computer Society seeks applicants for the position of editor in chief for the following publications for two-year terms starting **1 January 2009**: *IEEE Intelligent Systems, IEEE/ACM Transactions on Computational Biology and Bioinformatics, IEEE Transactions on Knowledge and Data Engineering*, and *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

QUALIFICATIONS AND REQUIREMENTS

Candidates for any Computer Society editor-in-chief position should possess a good understanding of industry, academic, and government aspects of the specific publication's field. In addition, candidates must demonstrate the managerial skills necessary to process manuscripts through the editorial cycle in a timely fashion. An editor in chief must be able to attract respected experts to his or her editorial board. Major responsibilities include

- actively soliciting high-quality manuscripts from potential authors and, with support from publication staff, helping these authors get their manuscripts published;
- identifying and appointing editorial board members, with the concurrence of the Publications Board;
- selecting competent manuscript reviewers, with the help of editorial board members, and managing timely reviews of manuscripts;
- directing editorial board members to seek special-issue proposals and manuscripts in specific areas;
- providing a clear, broad focus through promotion of personal vision and guidance where appropriate; and
- resolving conflicts or problems as necessary.

Applicants should possess recognized expertise in the computer science and engineering community and must have clear employer support.

SEARCH PROCEDURE

Prospective candidates are asked to provide, by **15 March**, a complete curriculum vitae, a brief plan for the publication's future, and a letter of support from their institution or employer. Materials should be sent as PDF files to the staff coordinators listed below.

Contact information for each publication follows.

- *IEEE Intelligent Systems*: www.computer.org/intelligent/eicsearch08.html
- *IEEE/ACM Transactions on Computational Biology and Bioinformatics*: Alicia Stickley, astickley@computer.org
- *IEEE Transactions on Knowledge and Data Engineering*: Alicia Stickley, astickley@computer.org
- *IEEE Transactions on Pattern Analysis and Machine Intelligence*: Alicia Stickley, astickley@computer.org

REAPPOINTMENTS

Several other IEEE Computer Society publications have editors in chief who are currently standing for reappointment to a second two-year term. The IEEE Computer Society Publications Board invites comments upon the tenures of the individual editors.

Editors in chief standing for reappointment to terms in 2009-2010 are Carl Chang, *Computer*; Fred Douglis, *IEEE Internet Computing*; David Albonesi, *IEEE Micro*; Carl Landwehr, *IEEE Security and Privacy*; Hakan Erdogmus, *IEEE Software*; Fabrizio Lombardi, *IEEE Transactions on Computers*; and Thomas Ertl, *IEEE Transactions on Visualization & Computer Graphics*.

Send comments to

- *Computer*: Scott Hamilton, shamilton@computer.org
- *IEEE Internet Computing*: Rebecca Deuel, rdeuel@computer.org
- *IEEE Micro*: Robin Baldwin, rbaldwin@computer.org
- *IEEE Security and Privacy*: Kathy Clark-Fisher, kclark-fisher@computer.org
- *IEEE Software*: Dale Strok, dstrok@computer.org
- *IEEE Transactions on Computers*: Alicia Stickley, astickley@computer.org
- *IEEE Transactions on Visualization & Computer Graphics*: Alicia Stickley, astickley@computer.org

NEW EDITORS IN CHIEF

Several other IEEE Computer Society publications have editors in chief who are beginning an initial two-year term in 2008.

Jeffrey Yost, of the University of Minnesota, now directs *IEEE Annals of the History of Computing*. Alan Clements, of the UK's University of Teesside, now leads the IEEE Computer Society Press. Virgil Gligor, of the University of Maryland, now heads *IEEE Transactions on Dependable and Secure Computing*. J. Edward Colgate of Northwestern University and Liang-Jie Zhang of IBM's T.J. Watson Research Center will serve as inaugural editors in chief of new titles *IEEE Transactions on Haptics* and *IEEE Transactions on Services Computing*, respectively.

CALLS FOR ARTICLES FOR IEEE CS PUBLICATIONS

IEEE Intelligent Systems seeks articles for a September 2008 special issue on natural-language processing and the web. Articles should focus on innovative uses of the web as a large-scale, distributed, hyperlinked, and multilingual corpus, and on building state-of-the-art natural-language interfaces to search engines.

Submissions are due by **5 March**. Visit www.computer.org/intelligent to view the complete call for papers.

IEEE Pervasive Computing seeks articles for a December 2008 issue on environmental sustainability. The creation, use, and disposal of large quantities of pervasive technologies such as sensors and mobile devices have strong implications for resource consumption and waste production. Submissions should address design for technology reuse, repurposing, or lifetime extension; sensor network applications that support the efficient use or protection of natural resources; novel systems, devices, or interfaces that support stewardship of the natural environment; and resource-efficient system design, among other topics.

Articles are due by **23 June**. Visit www.computer.org/pervasive to view detailed author instructions and the complete call for papers.

CALLS FOR PAPERS

LICS 2008, IEEE Logic in Computer Science Symp., 24-27 June, Pittsburgh; Submissions due **7 Jan;** www2.informatik.hu-berlin.de/lics/lics08/cfp08-1.pdf

COMPSAC 2008, 32nd IEEE Int'l Computer Software and Applications Conf., 28 July–1 Aug, Turku, Finland; Submissions due **15 Jan:** www.compsac.org

SCC 2008, IEEE Int'l Conf. on Services Computing, 8-11 July, Hawai'i; Submissions due **28 Jan;** http://conferences.computer.org/scc/2008/cfp.html

ICWS 2008, IEEE Conf. on Web Services, 23-26 Sept, Beijing; Submissions due **7 Apr;** http://conferences.computer.org/icws/2008/call-for-papers.html

Music And Multimedia 2008, The Use of Symbols To Represent Music And Multimedia Objects, 8 Oct, Lugano, Switzerland; Submissions due **31 May;** www.cm.supsi.ch

Submission Instructions

The Call and Calendar section lists conferences, symposia, and workshops that the IEEE Computer Society sponsors or cooperates in presenting.

Visit www.computer.org/conferences for instructions on how to submit conference or call listings as well as a more complete listing of upcoming computer-related conferences.

CALENDAR

JANUARY 2008

7-10 Jan: HICSS 2008, Hawai'i Int'l Conf. on System Sciences, Waikoloa, Hawai'i; www.hicss.hawaii.edu/hicss_41/apahome41.html

FEBRUARY 2008

18-21 Feb: WICSA 2008, Working IEEE/IFIP Conf. on Software Architecture, Vancouver, Canada; www.wicsa.net

MARCH 2008

3-7 Mar: SimuTools 2008, 1st Int'l Conf. on Simulation Tools and Techniques for Communications, Networks, and Systems, Vancouver, Canada; www.simutools.org

25-28 Mar: AINA 2008, 22nd IEEE Int'l Conf. on Advanced Information Networking and Applications, Okinawa, Japan; www.aina-conference.org/2008

25-28 Mar: SOCNE 2008, 3rd IEEE Workshop on Service-Oriented Architectures in Converging Networked Environments (with AINA), Okinawa, Japan; www.c-lab.de/RLS/SOCNE08

31 Mar - 4 Apr: ECBS 2008, 15th IEEE Int'l Conf. on Eng. of Computer-Based Systems, Belfast, Northern Ireland; www.compeng.ulster.ac.uk/events/ecbs2008

APRIL 2008

7-12 Apr: ICDE 2008, 24th IEEE Int'l Conf. on Data Engineering, Cancun, Mexico; www.icde2008.org

8-12 Apr: MCN 2008, 2nd IEEE Workshop on Mission-Critical Networking (with InfoCom), Phoenix; www.criticalnet.org

14 Apr: HiCOMB 2008, 7th IEEE Int'l Workshop on High-Performance Computational Biology (with IPDPS), Miami; www.hicomb.org

14-15 Apr: RAW 2008, 15th Reconfigurable Architectures Workshop (with IPDPS), Miami; www.ece.lsu.edu/vaidy/raw

14-18 Apr: IPDPS 2008, 22nd IEEE Int'l Parallel and Distributed Processing Symp., Miami; www.ipdps.org

Events in 2008

FEBRUARY

18-21 . . . . . .WICSA 2008

MARCH

3-7 . . . . . SimuTools 2008
25-28 . . . . . .AINA 2008
25-28 . . . . . SOCNE 2008
31 Mar-4 Apr . . . . . ECBS 2008

APRIL

7-12 . . . . . ICDE 2008
8-12 . . . . . .MCN 2008
14 . . . . . HiCOMB 2008
14-15 . . . . . RAW 2008
14-18 . . . . . IPDPS 2008
15-17 . . . . . InfoCom 2008
18 . . . . . .Hot-P2P 2008
18 . . . . . PCGrid 2008
18 . . . . . .SSN 2008

15-17 Apr: InfoCom 2008, 27th IEEE Conf. on Computer Communications, Phoenix; www.ieee-infocom.org

18 Apr: Hot-P2P 2008, 5th Int'l Workshop on Hot Topics in Peer-to-Peer Systems (with IPDPS), Miami; www.disi.unige.it/hotp2p/2008/index.php

18 Apr: PCGrid 2008, 2nd Workshop on Desktop Grids and Volunteer Computing Systems (with IPDPS), Miami; http://pcgrid.lri.fr

18 Apr: SSN 2008, 4th Int'l Workshop on Security in Systems and Networks (with IPDPS), Miami; www.cse.buffalo.edu/~fwu2/ssn08

MAY 2008

4-8 May: VLSI 2008, 28th IEEE VLSI Test Symp., San Diego; www.tttc-vts.org

5-7 May: ISORC 2008, 11th IEEE Int'l Symp. on Object/Component/Service-oriented Real-Time Distributed Computing, Orlando, Florida; http://ise.gmu.edu/isorc08

7-9 May: EDCC 2008, 7th European Dependable Computing Conf., Kaunas, Lithuania; http://edcc.dependability.org

12-13 May: HST 2008, 8th IEEE Int'l Conf. on Technologies for Homeland Security, Waltham, Massachusetts; www.ieeehomelandsecurityconference.org

14-16 May: ICIS 2008, 7th IEEE Int'l Conf. on Computer and Information Science, Portland, Oregon; http://acis.cps.cmich.edu:8080/ICIS2008

18-22 May: CCGrid 2008, 8th IEEE Int'l Symp. on Cluster Computing and the Grid, Lyon, France; http://ccgrid2008.ens-lyon.fr

22-24 May: ISMVL 2008, 38th Int'l Symp. on Multiple-Valued Logic, Dallas; http://engr.smu.edu/ismvl08

24 May: ULSI 2008, 17th Int'l Workshop on Post-Binary ULSI Systems (with ISMVL), Dallas; http://engr.smu.edu/ismvl08

JUNE 2008

4-6 June: SMI 2008, IEEE Int'l Conf. on Shape Modeling and Applications, Stony Brook, New York; www.cs.sunysb.edu/smi08

10-13 June: ICPC 2008, 16th IEEE Int'l Conf. on Program Comprehension, Amsterdam; www.cs.vu.nl/icpc2008

11-13 June: SIES 2008, IEEE 3rd Symp. on Industrial Embedded Systems, La Grande Motte, France; http://www.lirmm.fr/SIES2008

23-25 June: CSF 2008, 21st IEEE Computer Security Foundations Symp. (with LICS), Pittsburgh; www.cylab.cmu.edu/CSF2008

23-25 June: WETICE 2008, 17th IEEE Int'l Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises, Rome; www.sel.uniroma2.it/wetice08/venue.htm

23-26 June: ICITA 2008, 5th Int'l Conf. on Information Technology and Applications, Cairns, Australia; www.icita.org

24-27 June: LICS 2008, IEEE Symp. on Logic in Computer Science, Pittsburgh; www2.informatik.hu-berlin.de/lics/lics08

JULY 2008

7-11 July: Services 2008, IEEE Congress on Services, Hawai'i; http://conferences.computer.org/services/2008

8-11 July: CIT 2008, IEEE Int'l Conf. on Computer and Information Technology, Sydney; http://attend.it.uts.edu.au/cit2008

8-11 July: SCC 2008, IEEE Int'l Conf. on Services Computing, Hawai'i; http://conferences.computer.org/scc/2008

11-12 July: NCA 2008, 7th IEEE Int'l Symp. on Network Computing and Applications, Cambridge, Massachusetts; www.ieee-nca.org

28 July–1 Aug: COMPSAC 2008, 32nd IEEE Int'l Computer Software and Applications Conf. (with SAINT), Turku, Finland; www.compsac.org

28 July–1 Aug: SAINT 2008, IEEE/IPSJ Symp. on Applications and the Internet (with COMPSAC), Turku, Finland; www.saintconference.org

SEPTEMBER 2008

1-3 Sept: AVSS 2008, 5th IEEE Int'l Conf. on Advanced Video and Signal-Based Surveillance, Santa Fe, New Mexico; www.cpl.uh.edu/avss2008

23-26 Sept: ICWS 2008, IEEE Int'l Conf. on Web Services, Beijing; http://conferences.computer.org/icws/2008

IPDPS 2008

In 2008, the IEEE Computer Society's Technical Committee on Parallel Processing is launching a new event, the PhD Forum, as part of its flagship conference, the International Parallel & Distributed Processing Symposium.

The new forum aims to help students establish contacts for entering the job market while giving representatives from industry and academia a preview of developing technologies in parallel and distributed computing. To participate, students will be asked to submit a two-page extended abstract by **15 January**, describing the novelties and advantages of their thesis work. The top submissions will receive travel awards sponsored by the TCPP.

IPDPS 2008, which takes place in Miami from 14-18 April, marks the 22nd year of an event in computer science that attracts top computer engineers and scientists from around the world.

The five-day program includes contributed papers, invited speakers, panels, tutorials, and commercial participation, framed by workshops held on the first and last days.

For further details, visit the IPDPS web site at www.ipdps.org.

Call for Volunteers

CSAB (www.csab.org) is accepting applications from volunteers from the academic, private, and public sectors to serve as program evaluators in the ABET (www.abet.org) accreditation process for programs at the baccalaureate level in the academic disciplines over which CSAB has lead society responsibilities. Representing the IEEE Computer Society, the ACM, and the Association for Information Systems, CSAB is the ABET lead society for computer science, information systems, software engineering, and information technology.

For more information, contact Patrick M. LaMalva, executive director of CSAB (lamalva@csab.org; phone: +1 203 975 1117).

Reaching the Goal: How Managers Improve a Services Business Using Goldratt's Theory of Constraints, John Ricketts. Managing services is extremely challenging, making traditional industrial-management techniques inadequate. This book presents a breakthrough management approach that embraces what makes services different: their diversity, complexity, and unique distribution methods.

The author draws on Eli Goldratt's Theory of Constraints, one of this generation's most successful management methodologies, and thoroughly adapts it to the needs of today's professional, scientific, and technical services businesses.

This book's practical techniques reflect several years of advanced IBM research and consulting with enterprise clients. Step-by-step, the author shows how to apply these techniques throughout the most crucial business functions, from project management to finance and from process improvement to sales and marketing.

IBM Press; www.ibmpressbooks.com; 0-13-233312-0; 400 pp.

Model-Driven Testing Using the UML Testing Profile, P. Baker, Z.R. Dai, J. Grabowski, Ø. Haugen, I. Schieferdecker, and C. Williams. Written by the original members of an industry standardization group, this book describes a systematic, model-driven test process in the context of UML. It shows readers how to use UML to test complex software systems and provides a definitive reference for the only UML-based test specification language.

Readers will learn how to use UTP concepts for functional and nonfunctional testing, with sample applications and best practices for user interfaces and service-oriented architectures. Model-driven development has become an important new paradigm in software development and has already demonstrated considerable impact in reducing time to market and improving product quality. However, developing high-

quality systems requires not only systematic development processes but also systematic test processes.

Springer; www.springer.com; 978-3-540-72562-6; 183 pp.

Open Source: Technology and Policy, Fadi P. Deek and James A.M. McHugh. The open source movement is a worldwide effort to promote an open style of software development more aligned with the accepted intellectual style of science than the proprietary modes of invention characteristic of modern business. The idea is to keep the scientific advances created by software development openly available for everyone to use, understand, and improve. The very process of open source creation is highly transparent.

This book addresses prominent projects in the open source movement, along with its enabling technologies, social characteristics, legal issues, business venues, and public and educational roles.

Cambridge University Press; www.cambridge.org; 978-0-521-70741-1; 352 pp.

Advances in Applied Self-Organizing Systems, Mikhail Prokopenko, ed. Designers of self-organizing systems now face the challenge of validating and controlling nondeterministic dynamics. Overengineering the system might completely suppress self-organization with an outside influence, eliminating emergent patterns and decreasing robustness, adaptability, and scalability.

This book presents the state of the practice in engineering self-organizing systems and examines ways to balance design and self-organization in the context of applications. As demonstrated throughout, finding this balance helps developers deal with diverse practical challenges. Many algorithms proposed and discussed in this volume are biologically inspired. Readers will also gain an insight into cellular automata, genetic algorithms, artificial immune systems, snake-like locomotion, ant foraging, bird flocking, and mutualistic biological ecosystems, among others. Demonstrating the practical relevance and applicability of self-organization, this book might be of interest to advanced students and researchers in a wide range of fields.

Springer; www.springer.com; 978-1-84628-981-1; 376 pp.

Made to Break, Giles Slade. This book provides a history of 20th century technology as seen through the prism of obsolescence. America invented everything that is now disposable, the author notes, explaining how disposability was in fact a necessary condition for America's rejection of tradition and its acceptance of change and impermanence.

The author reveals the ideas behind obsolescence at work in such American milestones as the inventions of branding, packaging, and advertising, as well as the struggle for a national communications network, the development of electronic technologies and, with it, the avalanche of electronic consumer waste that could overwhelm America's landfills and poison its water within the coming decade.

History reserves a privileged place for societies that built things to last—forever, if possible. What place will it hold for a society addicted to consumption, a whole culture made to break? This book gives a detailed and harrowing picture of how, by choosing to support ever-shorter product lives, we might well be shortening our future way of life as well.

Harvard University Press; www.hup.harvard.edu; 0-674-02572-5; 336 pp.

Send book announcements to newbooks@computer.org.

The Public Eye

David Alan Grier
George Washington University

Computer security shares the methods and goals of computer science as a whole but has a couple of features that set it apart.

Over the years, Ray has made peace with his ignorance. That task has not been easy, as he started his career filled with enthusiastic naiveté. During his college career, he had become interested in computer security. He had read Clifford Stoll's classic *Cuckoo's Egg* (Pocket, 2000) about an early hacker infiltration of the computers at Lawrence Berkeley Laboratories and had followed the story of the 1988 Internet worm.

During that time, Ray found one of the few books on security and tested the problems that it described on a Unix workstation in somebody's laboratory. He downloaded the password file, gained access to system commands, and even modified parts of the operating system, finishing the work with the confidence of youth and an eagerness to do battle with the dark shadows of the human soul.

Over the next five or six years, Ray became an investigator of computer invasions, a detective who defended financial services industry computers against cybercrime. The work took him to three continents and exposed him to a large range of machines, both big and small. It gave him a badge that was heavy enough to stop a .45-caliber slug before it grazed the press of a dress shirt and a fist full of stories you might not want to share with your mother.

"It is kind of like being a firefighter," he explained. We were sitting in a coffee shop near my office that was filled with young men and women who were drinking designer caffeinated beverages and appeared deeply involved in their own conversations. "When the alarm bell sounds, I jump into my boots, leap onto the screaming engine and boldly engage the fire wherever it may be."

Initially, the work had been fun and exciting, but the experience started to drain him. "Our systems are under attack all the time," he explained, "and the attackers are not good people." His eyes were agitated and his face looked tired. This was not the Ray I had known. "Sure, the big systems are secured, but there are hundreds of ways the hackers can get into a computer. They can break into a contractor's workstation. They can exploit the weaknesses of some local application. We just don't know all the problems that are out there."

A FIRSTHAND EXPERIENCE

At the time, Ray's words had the ring of truth. I had recently seen the security weaknesses in a new financial system that our university had purchased. Despite my protests, I was told that I had to be trained on the system. The class was run by a perky young person, the kind of individual who finds inspiration in those corporate motivational posters that feature bold lighting, dramatic landscapes, and slogans like "Dare to exceed."

The training session had been boring beyond measure, as we had to be introduced to the "system philosophy" long before we could be taught any useful commands. Being unengaged and sitting at a workstation near the back of the room, I had starting fiddling with the commands. I easily found that I could get to the help menu, and from that menu, I could get to the operating system command line and elevate my status to that of a system administrator. I filled the next two hours by e-mailing friends, notifying the system administrator of the security hole in his system, and sending messages to the trainer's screen at the front of the room. Unsure of the source of my messages, the trainer did the only logical thing: She ignored them.

"You never know how you are going to get hit," Ray had said at the time. "It might come from inside the organization. It might come from outside. It might be something that you have never even heard of. My advice is to get the strongest protection software that you can find, run it all the time, and expect that you will be invaded anyway."

Ray had ended that conversation with little peace in his heart. Part of his concern might have come from the fact that he was changing jobs and did not know what the future held. In part, it might also have come from the doubt that eats away at a career, the doubt that suggests that you've accomplished little or done nothing of value in

spite of strong efforts and good intentions.

COMPUTER-SECURITY FEATURES

Computer security shares the methods and goals of computer science as a whole but has a couple of features that set it apart. First, it presents an unknown target to practitioners like Ray. It's one thing to prepare for uncertainties in business, changes in the regulatory environment, or new operational methods. It's quite another to brace for a new attack by novel software from an unknown quarter.

Second, computer security has a vulnerable position in the public mind. When done well, it is invisible. When done badly, "it calls the entire field of computer security into question," according to a pair of critics, "because the public can't distinguish between valid methods [of security] and those that have yet to be proven."

Over the years, Ray came to the conclusion that the practice of computer security was hampered by a third problem: raw embarrassment. Time and again, he would arrive at a site, flash his badge to the system administrator, and be told that there was nothing to see. The staff couldn't find the infected machine. Or they had erased the computer's log, cleaned its disk, and restored the machine to operating condition. As a group, they were often unwilling to admit that they had been attacked by anything: a virus, a worm, or a hacker.

Occasionally, the administrators were overcome by the reasoned fear that news of the attack would undermine their investors' confidence. It was an idea they used to justify their actions to themselves and their superiors, but it was the kind of lie that has driven the tragic tales of the past 50 years. We are far more likely to be destroyed by hiding an ill deed than by actually committing that deed. If Richard Nixon taught us nothing more, he showed the perils of the cover-up.

In the course of his work, Ray discovered that computer administrators, as often as not, were motivated by unreasoned fears, by ideas unsustained by reasonable logic. System managers told him that they resisted reporting intrusions because the culprit might be someone beyond the reach of the law, such as a juvenile or a foreign national or, in the worst of all possible scenarios, someone would gain a better position or more business from the notoriety of the incident.

Computer security has a vulnerable position in the public mind.

COMPUTER VULNERABILTY

Such concerns do have some foundation in reality. "The business side of investigating cybercrime is that it really needs to either put someone in jail or provide network intelligence that can be used to better protect systems on a large scale," Ray told me recently. We had gathered again at my favorite coffee shop to talk about hacking, denial-of-service attacks, and other challenges to the modern computer system.

A recent incident in Europe had confirmed Ray's historic pessimism about the vulnerability of computers. A network of zombie computers had overwhelmed the machines of an entire country. In spite of this news, Ray seemed remarkably relaxed. Rather than being frightened by an event that many had called the first example of cyberwarfare, he seemed at peace with the world.

"It's not that things are any better," he said, "it's probably that I'm just more used to it. The operating systems are better, which is nice, but the hackers have more tools at their disposal. Once, you needed to understand a system's actual source code to exploit it. Now, there are software tools that will help you write a worm, create a virus, or even probe an operating system's weaknesses. There are even libraries of exploit code for the most popular systems.

"Even with all the changes in technology," he added, "each call is more likely to be a porn scare than a real threat against a system."

"Porn scare?" I noted. I must admit that I looked around the room to see if anyone was listening to our conversation.

"You have to know your porn," he said. "We regularly get phone calls claiming that someone is running a child pornography website on an office computer. Ninety-nine times out of 100, these reports lead us to an ordinary commercial porn site that is siphoning money from some poor fool's credit card, and there is nothing illegal about that. But even to possess kiddie porn is a felony, so I run through a pornography decision tree to see if we need to investigate."

"A porn decision tree?" I asked, perhaps indiscreetly.

"Yup," Ray replied. "I don't want to inspect every piece of alleged porn to determine if it is legal, so I created a little series of yes/no questions to reach a conclusion. 'Did you see pictures on the site? Are the people clothed? Do the girls have …'"

"I get the point," I interjected. To my eye, the couple next to us had started to take an interest in our discussion, but you can never really be certain about such things.

HARD-BITTEN DETECTIVE

We paused for a moment and then moved to other subjects: mutual acquaintances, old projects, the state of the world. As we talked, I realized that I would have to abandon one of my plans for portraying Ray and his work in this essay. At one point, I had hoped to describe him as a Dashiel Hammett character, a tough-guy detective who used short sentences, active verbs, and phrases that conveyed a deep cynicism with the world.

But Ray laughed, and tough-guy detectives don't laugh. He talked

about caring for his parents, and tough guy detectives don't acknowledge their parents. He discussed the new hardwood floors he was getting for his house, something that would never have entered the world of Dashiel Hammett.

Yet, Ray's job compares to that of the hard-bitten detective who relies on help from the informer, the unscrupulous cop, or women in distress. "It helps to make lots of friends in this business," he said, "especially friends from other shores." He has friends who will help preserve evidence, give him access to network logs without a warrant, point him toward a likely suspect.

"We just finished a case on a major worm attack," he said. "Once we got the IP addresses, we found that it had been released by three people, two in the Czech Republic and one in the US. Because we have a treaty with the Czech Republic, we were able to put the Czechs in jail for maliciously disrupting computer services." At this point he paused and grinned.

"What?" I asked.

"We were going to do the same thing with the American, but we found pictures on his machine, pictures of little naked kids in compromising positions. It's easier to prove possession of illegal materials than to connect a machine with a worm attack." "Besides," he added, "the prison term is longer. He went down for child porn."

With that, we turned to the subject of Estonia, the country that had recently been overwhelmed by malicious Internet traffic. The dates of the attack were suspicious and suggested that the Russian government or a friend of the Russian government had initiated the attack.

The attacks began shortly after the Estonian leaders removed a Russian statue from downtown Tallin and ended on the day Russians celebrate their victory in World War II. However, the Russian government took no responsibility for the incident, and the meager evidence in the case could point to any one of many possible instigators. Some experts noted that the attack might have been arranged for as little as US$100,000. It could have been arranged by the Russian Mafia, a wealthy Russian nationalist, or some thug who was trying to show how tough he was. No one really knew.

With this issue of *Computer*, this column will move in a new direction. Under its old title, "In Our Time," it examined the universal themes of computer science. As "The Known World," the column will focus on the current state of computing technology and the role of the various individuals who contribute to the field.

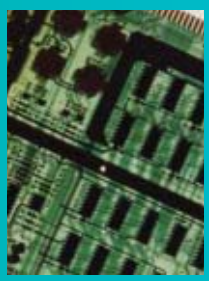
David Alan Grier *is an associate professor of International Science and Technology Policy at George Washington University and is the author of* When Computers Were Human *(Princeton University Press, 2005).*

A fuller discussion of the attacks on Estonia can be found in the July/Aug. 2007 issue of IEEE Security and Privacy, *"The New Front Line," by Michael Lesk (pp. 76-79.)*

IEEE Computer Society

2008 MEMBERSHIP APPLICATION





FIND THE RIGHT SOLUTION!

Solve problems, learn new skills, and grow your career with the cutting edge resources of the IEEE Computer Society.

www.computer.org/join

IEEE

IEEE computer society

2008 RATES for IEEE COMPUTER SOCIETY

Membership Dues and Subscriptions

Membership and periodical subscriptions are annualized to and expire on 31 December 2008.
Pay full or half-year rate depending upon the date of receipt by the IEEE Computer Society as noted below.

Membership Options*

All prices are quoted in U.S. dollars.

	FULL YEAR Applications received 17 Aug 07 – 29 Feb 08	HALF YEAR Applications received 1 Mar 08– 15 Aug 08
I do not belong to the IEEE and I want to join only the Computer Society:	☐ $113.00	☐ $57.00
I want to join **both** the Computer Society and the IEEE:		
I reside in the USA	☐ $220.00	☐ $110.00
I reside in Canada	☐ $195.00	☐ $98.00
I reside in Africa/Europe/Middle East	☐ $187.00	☐ $94.00
I reside in Latin America	☐ $180.00	☐ $90.00
I reside in Asia/Pacific	☐ $181.00	☐ $91.00
I already belong to the IEEE, and I want to join the Computer Society:	☐ $50.00	☐ $25.00

Are you now or were you ever a member of the IEEE?
☐ Yes ☐ No If yes, please provide member # if known: _____

Add Periodicals**

	ISSUES PER YEAR	FULL YEAR Applications received 16 Aug 07 – 29 Feb 08 PRINT + ONLINE	HALF YEAR Applications received 1 Mar 08 – 15 Aug 08 PRINT + ONLINE
BEST VALUE!			
IEEE Computer Society Digital Library (online only)	n/a	☐ $121	☐ $61
ARTIFICIAL INTELLIGENCE			
IEEE Intelligent Systems	6	☐ $43	☐ $22
IEEE Transactions on Learning Technologies†	4	☐ $39	☐ $18
IEEE Transactions on Pattern Analysis and Machine Intelligence	12	☐ $52	☐ $26
BIOTECHNOLOGY			
IEEE/ACM Transactions on Computational Biology and Bioinformatics	4	☐ $36	☐ $18
COMPUTATION			
Computing in Science & Engineering	6	☐ $45	☐ $23
COMPUTER HARDWARE			
IEEE Computer Architecture Letters	4	☐ $29	☐ $15
IEEE Micro	6	☐ $41	☐ $21
IEEE Design & Test of Computers	6	☐ $40	☐ $20
IEEE Transactions on Computers	12	☐ $47	☐ $24
GRAPHICS & MULTIMEDIA			
IEEE Computer Graphics and Applications	6	☐ $43	☐ $22
IEEE MultiMedia	4	☐ $38	☐ $19
IEEE Transactions on Haptics†	2	☐ $31	☐ $16
IEEE Transactions on Visualization and Computer Graphics	6	☐ $43	☐ $22
HISTORY OF COMPUTING			
IEEE Annals of the History of Computing	4	☐ $34	☐ $17
INTERNET & DATA TECHNOLOGIES			
IEEE Internet Computing	6	☐ $43	☐ $22
IEEE Transactions on Knowledge and Data Engineering	12	☐ $49	☐ $25
IEEE Transactions on Services Computing†	4	☐ $39	☐ $18
IT & SECURITY			
IT Professional	6	☐ $42	☐ $21
IEEE Security & Privacy	6	☐ $24	☐ $12
IEEE Transactions on Dependable and Secure Computing	4	☐ $33	☐ $17
MOBILE COMPUTING			
IEEE Pervasive Computing	4	☐ $43	☐ $22
IEEE Transactions on Mobile Computing	12	☐ $43	☐ $22
NETWORKING			
IEEE Transactions on Parallel and Distributed Systems	12	☐ $47	☐ $24
SOFTWARE			
IEEE Software	6	☐ $49	☐ $25
IEEE Transactions on Software Engineering	12	☐ $38	☐ $19

* Member dues include $25 for a 12-month subscription to *Computer* magazine.
** Periodicals purchased at member pirces are for the member's personal use only.
† Online issues only.

Payment Information

Payment required with application

Membership fee
$

Periodicals total
$

Applicable sales tax***
$

TOTAL:
$

Enclosed:
☐ Check/Money Order****

Charge my:
☐ MasterCard
☐ VISA
☐ American Express
☐ Diner's Club

Card Number

Exp Date (month/year)

Signature

USA Only include
5-digit billing zip code

* Member dues include $25 for a 12-month subscription to *Computer*.

** Periodicals purchased at member prices are for the member's personal use only.

*** Canadian residents add 14% HST or 6% GST to total. AL, AZ, CO, DC, GA, IN, KY, MD, MO, NM, and WV add sales tax to periodical subscriptions. European Union residents add VAT tax to IEEE Computer Society Digital Library subscription.

**** Payable to the IEEE in U.S. dollars drawn on a U.S. bank account. Please include member name and number (if known) on your check.

Allow up to 8 weeks for application processing. Allow a minimum of 6 to 10 weeks for delivery of print periodicals.

Please complete both sides of this form.

For fastest service, apply online at www.computer.org/join

Personal Information

Enter your name as you want it to appear on correspondence. As a key identifier in our database, circle your last/surname.

☐ Male ☐ Female

Date of birth (Day/Month/Year)

Title | First name | Middle

Last/Surname

Home address

City | State/Province

Postal code | Country

Home telephone

Home facsimile

Preferred e-mail

Send mail to: ☐ Home address ☐ Business address

Educational Information

First professional degree completed | Month/Year degree received

Program major/course of study

College/University | State/Province | Country

Highest technical degree received | Program/Course of study

Month/Year received

College/University | State/Province | Country

Business/Professional Information

Title/Position

Years in current position | Years of practice since graduation

Employer name

Department/Division

Street address

City | State/Province

Postal code | Country

Office phone

Office facsimile

I hereby make application for Computer Society and/or IEEE membership and agree to be governed by IEEE's Constitution, Bylaws, Statements of Policies and Procedures, and Code of Ethics. I authorize release of information related to this application to determine my qualifications for membership.

Signature | Date

IF8A

NOTE: In order for us to process your application, you must complete and return BOTH sides of this form to the office nearest you:

Asia/Pacific Office
IEEE Computer Society
Watanabe Bldg.
1-4-2 Minami-Aoyama
Minato-ku, Tokyo 107-0062 Japan
Phone: +81 3 3408 3118
Fax: +81 3 3408 3553
E-mail: tokyo.ofc@computer.org

Publications Office
IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314 USA
Phone: +1 800 272 6657 (USA and Canada)
Phone: +1 714 821 8380 (worldwide)
Fax: +1 714 821 4641
E-mail: help@computer.org

BPA Information

This information is used by society magazines to verify their annual circulation. Please refer to the audit codes and indicate your selections in the box provided.

A. Primary line of business

1. Computers
2. Computer peripheral equipment
3. Software
4. Office and business machines
5. Test, measurement, and instrumentation equipment
6. Communications systems and equipment
7. Navigation and guidance systems and equipment
8. Consumer electronics/appliances
9. Industrial equipment, controls, and systems
10. ICs and microprocessors
11. Semiconductors, components, sub-assemblies, materials, and supplies
12. Aircraft, missiles, space, and ground support equipment
13. Oceanography and support equipment
14. Medical electronic equipment
15. OEM incorporating electronics in their end product (not elsewhere classified)
16. Independent and university research, test and design laboratories, and consultants (not connected with a manufacturing company)
17. Government agencies and armed forces
18. Companies using and/or incorporating any electronic products in their manufacturing, processing, research, or development activities
19. Telecommunications services, and telephone (including cellular)
20. Broadcast services (TV, cable, radio)
21. Transportation services (airlines, railroads, etc.)
22. Computer and communications and data processing services
23. Power production, generation, transmission, and distribution
24. Other commercial users of electrical, electronic equipment, and services (not elsewhere classified)
25. Distributor (reseller, wholesaler, retailer)
26. University, college/other education institutions, libraries
27. Retired
28. Others (allied to this field) _____

B. Principal job function

1. General and corporate management
2. Engineering management
3. Project engineering management
4. Research and development management
5. Design engineering management — analog
6. Design engineering management — digital
7. Research and development engineering
8. Design/development engineering — analog
9. Design/development engineering — digital
10. Hardware engineering
11. Software design/development
12. Computer science
13. Science/physics/mathematics
14. Engineering (not elsewhere classified)
15. Marketing/sales/purchasing
16. Consulting
17. Education/teaching
18. Retired
19. Other _____

C. Principal responsibility

1. Engineering or scientific management
2. Management other than engineering
3. Engineering design
4. Engineering
5. Software: science/management/engineering
6. Education/teaching
7. Consulting
8. Retired
9. Other _____

D. Title

1. Chairman of the Board/President/CEO
2. Owner/Partner
3. General Manager
4. V.P. Operations
5. V.P. Engineering/Director Engineering
6. Chief Engineer/Chief Scientist
7. Engineering Manager
8. Scientific Manager
9. Member of Technical Staff
10. Design Engineering Manager
11. Design Engineer
12. Hardware Engineer
13. Software Engineer
14. Computer Scientist
15. Dean/Professor/Instructor
16. Consultant
17. Retired
18. Other Professional/Technical _____

Safety Issues in Modern Bus Standards

Janusz Sosnowski and Dawid Trawczyński,
Warsaw University of Technology
Janusz Zalewski, Florida Gulf Coast University

Selecting and designing bus standards for safety-critical applications requires careful analysis of error-detection and fault-handling mechanisms.

The development of computer buses—among the first computing elements to have been standardized—has been well documented in the literature by, for example, such titles as *Advanced Multimicroprocessor Bus Architectures* (J. Zalewski, ed., IEEE CS Press, 1995) and *Industrial Technology Communication Handbook* (R. Zurawski, ed., CRC, 2003). None of these older or newer standards pay much attention to dependability issues except for some signal lines reporting simple communication errors, but without any significant focus on recovery procedures.

With the advent of safety-critical systems, such as those used in avionics, nuclear, medical, automotive, and other regulated industries, it became evident that dealing with errors in bus design and usage is necessary. Bus designs must prevent any error from occurring and, if one does occur, developers must apply efficient error-detection and recovery procedures. This is required for qualification and certification of electronic devices used in safety-critical applications, such as steer-by-wire systems in cars or fly-by-wire systems on aircraft, following such standards as RTCA/DO-254, *Design Assurance Guidance for Airborne Electronic Hardware*.

The most recent buses, even though touted as being designed for high dependability and safety-critical applications, require thorough testing of their protocols under load to gain users' confidence in the designers' dependability claims.

FAULT EFFECTS

Bus dependability and safety relate to permanent physical faults such as hardware damage; intermittent faults such as temporal discrepancies, specific defects that reveal randomly in time; and transient faults caused by phenomena such as EM noise and radiation. These faults can result in observed logical errors and cause operational failures.

In bus design, the following failure types should be considered: masquerading, babbling idiot, slightly-off specification, and spatial proximity, among others such as outgoing link failure.

A masquerading failure occurs if an erroneous node imitates another node's operation. Developers can eliminate this by systematically checking node states and verifying the received messages with the state consistency of both the receiver and sender. A babbling-idiot node, which can interfere with correctly operating nodes, tries to send messages at incorrect times. Bus guardians can eliminate this.

Slightly-off specification failures usually occur at the border of analog and digital circuitry. If an erroneous node produces an output signal slightly outside the specified acceptance window in time or value, some nodes will correctly receive the signal while others might fail to receive it. This results in the distributed system experiencing an inconsistent state, often called a Byzantine fault.

A spatial proximity failure occurs if the replicated units are in close physical proximity, in which case a single external event such as overheating might disable the replicated units. Developers can avoid this by using a star configuration in which all units are dispersed and wired to a central location, usually a hub.

BUS DESIGN FOR SAFETY

Because they are statistics-based, traditional parameters for bus evaluation, such as data throughput and latency, although important in bus design, might not give the best estimate of bus quality in safety-critical applications. In such systems, bus evaluation requires using more than just statistical criteria because bus behavior must adhere to more stringent requirements in such applications. The bus protocol must guarantee freedom from failures under predictable circumstances, and when unpredictable failures occur,

0018-9162/08/$25.00 © 2008 IEEE Published by the IEEE Computer Society

it must ensure an orderly recovery. Thus, developers must take a significantly different approach when designing buses for safety-critical applications.

From the certification perspective, the relevant literature has addressed three essential aspects of the risk-assessment process: bus evaluation criteria and metrics, hazard analysis, and failure-mode analysis. The Certification Authorities Software Team's Position Paper #16 proposed the initial criteria selection (www.faa.gov/certification/aircraft/av-info/software/CAST/cast-16.rtf).

The major issues to consider when assessing bus operation include safety, data integrity, performance, design and development assurance, and validation and testing approaches. A specific array of tests can demonstrate data integrity and performance. The process should define the allowed error rate per byte and provide the means to recover from errors. It should also specify the load analysis and related bus capacity. The analyses and tests should consider the extreme cases of bus loss, line shorts, and breaks.

When designing dependable and safety-critical systems, developers must take into account the defined fault classes and describe procedures for dealing with faults that will make the system more fault tolerant. They also must consider system reaction to faults outside the specified class, such as assuring a safe state or designing a system so that there is a very low probability that unaccounted-for faults, such as correlated multiple transient faults, will occur. For example, using duplicated transmission channels with different media helps avoid correlated faults.

Developers must also ensure that the system is functionally and physically partitioned into blocks—effectively creating error-containment areas—in such a way that they can detect and correct or mask the consequences of faults in one block before those consequences corrupt the rest of the system.

Such partitioning also simplifies fault diagnosis. Dependable systems must report a node's failure consistently to all operating nodes within an acceptable period. This requires a special protocol to establish the set of operational nodes that remain aware of other nonfaulty nodes' operation. In complex situations, the system can create different cliques comprising different node groups, considered nonfaulty. Reducing the probability that a single erroneous node interferes continuously with the proper operation of another node requires the initiation of specific actions.

The traditional approach to evaluating a bus design measures its two most important performance parameters: data throughput and data latency.

Many applications demand that the system guarantee receipt of messages without exceeding preset delay limits, while also accounting for jitter. Usually, four sources of jitter manifest at the bus level:

- bit stuffing (synchronization of bit-level requirements),
- task scheduling (variations in time to actually execute software tasks in a node),
- interference of periodic messages (time triggered), and
- higher-priority messages arriving in unpredictable times (event triggered).

Hence, sound bus designs have to address and minimize the consequences of such jitter.

SELECTED BUS DESIGNS

Bus applications and case studies presented in the literature, such as steer-by-wire and fly-by-wire systems, give a broader context for developing bus-safety evaluation criteria. For general aviation aircraft such as business jets and smaller aircraft, and for automotive applications as well, several different communications technologies have been developed, including CAN (controller area network; www.iso.org), FlexRay (www.flexray.com), the Time-Triggered Communication Protocol (www.vmars.tuwien.ac.at/projects/ttp/ttpc.html), and SAFEbus (ARINC 659; www.arinc.com).

In most buses, embedded error-detection mechanisms cover frame format control, data errors (CRC codes with correction capabilities), control flow checking with acknowledgments, and so on. CAN and TTCAN (time-triggered controller area network; www.ttcan.com) use fault counters to distinguish transient, intermittent, and permanent faults and to initiate appropriate recovery procedures. CAN is prone to jitter, which is reduced to 180 microseconds in FlexCAN by introducing transmission subcycles.

The TTP/C protocol assures duplicated nodes and buses, clique detection for all asymmetric communication faults, different dividing polynomial seeds for dual-channel operation, distributed clock synchronization with offset correction in the microseconds range, and so on. Safe-by-Wire (www.semiconductors.philips.com/acrobat/other/automotive/safe-by-wire_bus_spec_1.0.pdf) is immune to babbling-idiot errors, provides multilevel protection against inadvertent deployment, and tolerates shorts or breaks in bus wires.

EXPERIMENTAL BUS SAFETY EVALUATION

In principle, the traditional approach to evaluating a bus design measures its two most important performance parameters: data throughput and data latency. In safety-related applications, the main issue involves predicting when

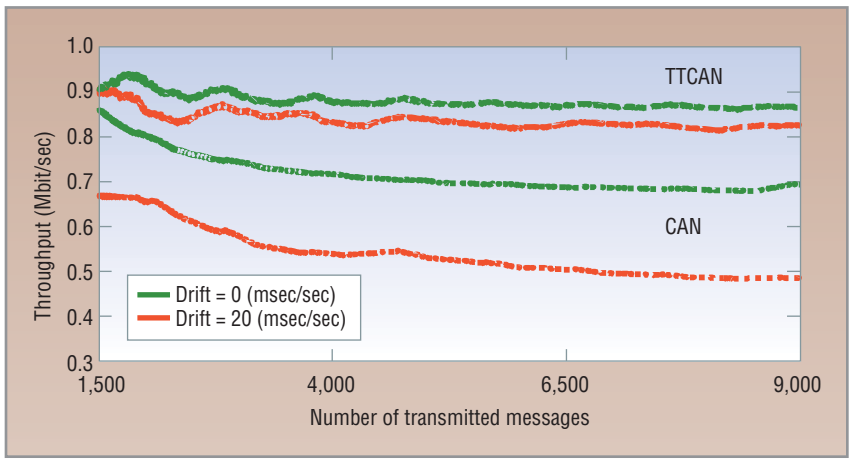


Figure 1. Impact of clock drift on CAN and TTCAN bus throughput. Simulations show that clock drift—20-ms drift during the whole one-second simulation period—has a larger negative impact on CAN than on TTCAN.

the bus might become the source of errors and when its failure might initiate or contribute to an event chain that would cause a hazard or danger, leading to unsafe behavior in an embedded system. Data throughput and latency address only bus performance, not the consequences of failures, which relate directly to safety. In this view, to study safety a bus should be placed in a broader context, for example:

- *system-oriented*, such as another network or different modes of operation, or
- *software-oriented*, such as a software driver or higher-level protocol.

Considering the issue of bus safety, we designed a series of simulation experiments with the ultimate objective of acquiring more comprehensive information on bus behavior than that available from straightforward performance evaluations. In particular, we sought to check the impact on system behavior of various critical situations and faults, the effectiveness of recovery procedures, and related factors.

For this purpose, the system design can support classical simulation tools with various fault injectors at the physical, logical, or software levels. For example, Figure 1 shows the results of the impact of the clock-drift fault on CAN and TTCAN throughput. We performed the experiment using the TrueTime simulator (www.control.lth.se/truetime) enhanced with our own fault-injection software.

The model network comprised eight nodes, all transmitting specific-length messages periodically. Each node retransmitted the received messages to its neighbor, so the traffic on the bus systematically increased.

The simulations show that the clock drift—20-ms drift during the whole one-second simulation period—has a larger negative impact on CAN than on TTCAN. The average throughput for CAN decreased by 30 percent compared to TTCAN, which only decreased by approximately 5 percent. Additionally, on average, the TTCAN bus arbitration protocol achieved throughput about 41 percent higher than that of CAN under the given fault and network load. In some applications, a significant decrease in throughput can be critical. The follow-up experiment would check jitter for typical and critical operational scenarios in the case of no faults and also when disturbed by a specific class of faults.

Selecting and designing bus standards for safety-critical applications requires careful analysis of error-detection and fault-handling mechanisms. This analysis must be based on the revision of standard specifications and an experimental evaluation that covers representative fault classes.

Janusz Sosnowski is a professor and chair in the Institute of Computer Science at the Warsaw University of Technology. Contact him at jss@ii.pw.edu.pl.

Dawid Trawczyński is a PhD candidate in the Institute of Computer Science at the Warsaw University of Technology. Contact him at dawid_trawczynski@yahoo.com.

Janusz Zalewski is a professor in the Department of Computer Science at Florida Gulf Coast University. Contact him at zalewski@fgcu.edu.

Editor: John Harauz, Jonic Systems Engineering, Inc., Willowdale, Ont., Canada; j.harauz@ieee.org

The Impact of Outsourcing on Client Project Managers

Mary C. Lacity and Joseph W. Rottman
University of Missouri-St. Louis

Offshore outsourcing of IT work brings both benefits and challenges.

Between 2004 and 2007, we interviewed 232 information technology practitioners on the topic of offshore outsourcing of IT. The participants included 24 US client organizations and 33 offshore suppliers based in India, China, and Canada.

The results of this study appear in our forthcoming book, *Offshore Outsourcing of IT Work* (Palgrave Macmillan, 2008), which details the client-side roles of chief information officers, program management officers, and client project managers and identifies characteristics that distinguish successful projects.

The study sample included interviews with 67 project managers. Whereas senior IT leaders design global outsourcing strategies, project managers are responsible for integrating supplier employees and delivering the promised value. We found that offshore outsourcing made their jobs easier in four ways but also created 20 "challenges"—our collective term for what they actually described as "problems," "headaches," or "crises."

OUTSOURCING BENEFITS

Client project managers reported four offshore outsourcing benefits.

Faster staffing

Most project managers initially welcomed offshore suppliers, whose deep pool of available talent made it possible to staff quickly. "Our Indian supplier is great at finding people," one project manager from a US retailer said. "Before them, I would be scrambling within [the retail company] trying to find more people. Nobody had anybody available. So, I can just go to our supplier and say, 'Send me three people.' And they are here."

Faster development

Some project managers were able to complete work more quickly by exploiting time-zone differences between their company and the offshore supplier. For example, one respondent built a large system in three months with the help of an Indian supplier instead of the estimated six months for internal development. He synchronized work so that the Indian employees worked on the project while US workers slept, and vice versa.

Access to scarce skills

Project managers were often delighted to have access to the offshore supplier's scarce technical abilities. One US financial services firm, for example, drew on 250 supplier employees to meet critical skill shortages in Java, Perl, and web-based development. "Even if it's a wash on cost savings," one company participant said, "I'd have a hard time finding and bringing in 250 employees here at headquarters."

Motivated workforce

Despite significant cultural differences between Western clients and offshore suppliers, nearly all project managers noted that supplier employees were intelligent, pleasant, and eager to please.

OUTSOURCING CHALLENGES

Despite their positive experiences working with offshore suppliers, client project managers were often unprepared for the changes required in their roles. Table 1 categorizes 20 major effects of offshore outsourcing by area of concern.

Organizational support

In best-practice organizations, clients have mature program management offices to govern offshore outsourcing relationships. Our book identifies 15 PMO roles needed to support project managers and ensure offshore outsourcing success.

Unfortunately, many PMOs we studied were woefully understaffed. Consequently, project managers often had to assume many of these roles, which distracted them from

Table 1. Twenty major effects of offshore outsourcing reported by project managers.

Area of concern	Effects
Organizational support	1. They had to fill many of the roles that the project management office should have performed.
	2. They needed a mentor the first time they managed a project with offshore resources.
Project planning	3. They needed to thoroughly verify the offshore supplier's work estimates, which tended to be optimistic.
	4. They experienced higher transaction costs, which threatened their ability to deliver projects on budget.
	5. They experienced project delays, which threatened their ability to deliver projects on time.
Knowledge transfer	6. They had to do more knowledge transfer up front.
	7. They were often forced to shortcut the knowledge transfer process because of deadlines set by senior IT leaders.
	8. They had to ensure that knowledge transfer was successful by testing the supplier employees' knowledge.
	9. They had to guarantee that the supplier followed preagreed knowledge renewal practices.
	10. They had to ensure that the supplier transferred knowledge about new applications or technologies to the client.
	11. They had to learn about new applications or technologies independent of suppliers to guarantee that the supplier's information and bids were valid.
CMM/CMMI processes	12. They had to provide greater detail in requirement definitions.
	13. They had to integrate the supplier's CMM/CMMI processes into their own project-management processes.
	14. They had to ensure that the supplier's employees were fully trained as promised by suppliers.
Managing work	15. They had to set more frequent milestones.
	16. They needed more frequent and more detailed status reports.
	17. They required more frequent working meetings to prevent client-caused bottlenecks.
Managing people	18. They had to motivate the supplier to share bad news.
	19. They needed to accompany offshore suppliers to all client-facing meetings.
	20. They had to make offshore suppliers feel welcome and comfortable.

their other duties and sometimes became overwhelming. For example, many observed that project launches were delayed by internal structural issues they assumed the PMO had addressed. The most frequently cited problems were visa delays and the inability to provide offshore personnel secure access to client systems and remote data.

Project planning

Although project managers often negotiate plans with business sponsors, capital budgeting committees, IT planning committees, and suppliers, they're ultimately responsible for delivering those projects on time, within budget, and with promised functionality and quality. The inclusion of offshore suppliers affected project planning in three major ways.

First, project managers had to more thoroughly verify the work estimates of offshore suppliers, who often didn't fully understand requirements or were overly optimistic. Some project managers frankly told offshore suppliers, "This estimate is too low," and emphasized that they wanted the "most likely" and not the "most optimistic" forecast. Several simply increased time estimates by 30 to 50 percent.

Second, project managers experienced significant hidden transaction costs. For example, several said they had falsely assumed that suppliers held licenses for most software products. The additional software license fees were tacked onto their budgets, which proved quite costly on large projects with 50 or more offshore supplier employees.

Third, projects were delayed because of personal events such as weddings and births, and national events such as elections and holidays, which can take much longer in Eastern cultures than in Western ones.

Knowledge transfer

Client project managers had to learn new ways to transfer knowledge to and from offshore suppliers.

When the team included only internal IT staff and domestic contractors, project managers transferred knowledge incrementally. However, when a project included offshore employees, knowledge transfer occurred in a more concentrated time frame. Some members of the offshore delivery team were only on site for a few weeks, so the project managers planned for intensive knowledge transfer.

Further, project managers had to ensure that knowledge transfer was successful by testing supplier knowledge because offshore employees rarely expressed incomprehension. Some project managers asked their offshore contractors detailed questions to make sure they understood the business requirements.

To protect their knowledge investment, many client organizations included a contractual clause that required the supplier to have replacements shadow incumbent employees for two to four weeks, depending on the nature of the work. However, project managers often had no good way to verify that this actually occurred, particularly on large projects. A few

suspected that new hires were assigned to projects and billed to clients before the required shadowing period took place.

Knowledge renewal was also a challenge because of high supplier turnover.

CMM/CMMI processes

Offshore suppliers took pride in achieving high Capability Maturity Model or Capability Maturity Model Integrated levels, which to them signaled quality. However, the project managers we interviewed expressed real skepticism about suppliers' true commitment to CMM/CMMI.

One project manager, for example, said her supplier bragged about its CMM processes during sales and negotiations, but the employees assigned to her team were slow to respond when asked to show their code reviews, inspections, and test cases, and these were of inferior quality. After much probing, she learned that the supplier assigned new hires to her account before they had completed their advertised "intensive CMM training."

CMM/CMMI also required project managers to provide unexpectedly detailed written specifications to their offshore suppliers. One project manager was surprised when a financial statement came back with the dollar fields left-justified. The supplier responded, "You didn't say you wanted them right-justified."

Project managers who didn't plan enough time up front to detail requirements had to make extensive revisions that caused delays downstream. Some decided to use the supplier's templates, which provided a good idea of what the supplier employees needed and thereby limited rework.

Managing work

Client project managers were often told to manage the supplier's work products rather than its staff. However, offshore suppliers uniformly failed to report when they were going to miss a deadline, making it difficult for project managers to trust the supplier to independently complete a packet of work.

To better manage the supplier's work products, project managers created more frequent milestones. They typically required domestic suppliers to produce a major milestone every two months, but many required offshore suppliers to deliver milestones every two weeks.

The project managers we interviewed expressed real skepticism about suppliers' true commitment to CMM/CMMI.

Project managers also required more detailed status reports. At one US bank, for example, the manager created an online form with specific questions to make it easier for the offshore team lead to report delays in written form.

Finally, project managers called more frequent working meetings to prevent client-caused bottlenecks. Offshore programmers often halted work because they were waiting for the client to answer a question, finish a database schema, approve a deliverable, or define a requirement.

Managing people

In addition to assigning concrete tasks to offshore teams, project managers had to learn what each supplier employee did to verify supplier invoices on staff augmentation engagements. They also had to manage the user-supplier relationship, welcoming and integrating onsite supplier employees as well as accompanying them to user meetings to prevent "scope creep."

Without the client project manager's presence, users requested many new features, and suppliers were happy to comply. One participant said: "Scope creep? It was scope explosion! If the user wants it, then that's a new project or something to that effect. Because the supplier is so willing to do things and so willing to please—that's their culture—we were finding that they were doing things that we couldn't afford. Now, even though they may go to user meetings, there's always an IT person there."

In addition, some users complained to the project managers about speaking directly to the offshore staff. By accompanying suppliers to client-facing meetings, project managers or their designees served important boundary-spanning roles.

Regardless of the many challenges client project managers experienced with offshore outsourcing, they reported that it provided unique opportunities for professional and personal growth. Professionally, the ability to successfully manage globally dispersed teams is a skill employers value and recognize in terms of promotion and compensation. Personally, many client managers who traveled to developing countries for the first time made lasting friendships with supplier employees.

Mary C. Lacity *is a professor of information systems in the College of Business Administration at the University of Missouri-St. Louis. Contact her at mary.lacity@umsl.edu.*

Joseph W. Rottman *is an assistant professor of information systems in the College of Business Administration at the University of Missouri-St. Louis. Contact him at rottman@umsl.edu.*

Editor: Richard G. Mathieu, Dept. of Computer Information Systems and Management Science, College of Business, James Madison Univ., Harrisonburg, VA; mathierg@jmu.edu

Dependability and Security Will Change Embedded Computing

Dimitrios Serpanos, University of Patras and ISI, Greece
Jörg Henkel, University of Karlsruhe, Germany

A unified approach to dependability and security assessment will let architects and designers face the challenges of shrinking feature size and increasing error rates.

Embedded computing systems are found in everything from automobiles and communication devices to entertainment gadgets and industrial environments. Considering that users demand performance, reliability, and durability—delivered at low cost and with ease of management—we need to make significant progress in embedded systems' architecture, design, and development methodologies.

In a wide range of applications, embedded systems differ from general-purpose computing systems in their restricted resources (computation, communication, storage, and power) and their criticality (dependability, security, and safety) in applications and services.

Dependability is a well-defined term in computing systems. It means continuous operation in most environments, as well as predictable behavior in terms of functionality, performance, and timing (real-time requirements).

On the other hand, *security* means ensuring that the system achieves the expected, predictable behavior in protecting sensitive and private data, systems, and processes from attacks and accidents. The definition of *sensitive data* is critical in this context, because in safety-critical environments, we can classify as sensitive even the monitoring and auditing data needed to analyze processes and events after accidents.

Finally, *safety* means that the predictable behavior has specific safety properties under all circumstances. Clearly, safety properties differ significantly depending on the final process or service integrating the embedded systems. For example, safety requirements in automotive systems are quite different from safety requirements in oil refineries.

Thus, safety is a process requirement. However, its implementation depends heavily on dependability and security mechanisms to provide the properties and meet the requirements set for process safety. For example, continuous operation, a dependability property, is a necessity for safety-critical applications.

Considering that, we can view each embedded system in a networked world as a stack of functions, as Figure 1 shows. Instead of applying the seven layers of the OSI reference model, we present a compact stack composed of the

- node (Layer 1), which includes networking that offers basic processing, storage, and communication functions;
- dependability and security mechanisms (Layer 2), which enable the implementation of continuous operation, safety-critical, and fully functional applications at the subsequent layer; and
- safe process and application (Layer 3).

Following this approach, dependability and security mechanisms are peer mechanisms appearing at the middle layer of networked embedded systems, complementing each other to provide effective and efficient networked embedded systems.

DEPENDABILITY

Dependability is becoming an increasingly important issue, especially when considering the challenge that Moore's law imposes. For the past four decades Moore's law provided a situation in which all major design constraints benefited significantly. Higher integration and smaller feature sizes meant more functionality per chip, higher performance, and lower per-transistor cost.

However, the future holds a large bucket of challenging problems as feature size decreases and higher integration is achieved. When migrating to emerging technol-

0018-9162/08/$25.00 © 2008 IEEE

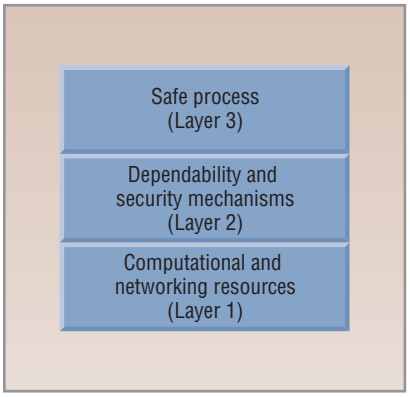


Figure 1. Embedded system layers. Each embedded system can be considered as a stack of functions.

ogy nodes of 45 nm and beyond, the inherent undependability at the physical and transistor level increases at an alarming rate.

Migrating to new technology nodes

Due to microminiaturization, designers can't control the fabrication process as they did in past technology nodes. As a result, the number of defective, on-chip devices (or modules) increases steeply, leading to far lower yields, defined as the number of flawless chips produced in relation to all chips in a certain batch. Using the current definition, yield will drop to zero, according to current predictions.

As miniaturization continues, even devices without flaws exhibit a wide variance of electrical properties due to process variation. One effect is *random dopant fluctuation*. As the number of dopant atoms decreases exponentially (only a few dozen dopant atoms are employed per transistor channel in 16-nm technology), any variation has a large impact. Process variations can lead to changing electrical properties such as increased delay times.

Furthermore, the susceptibility of future highly integrated circuits to *single-event upsets*—strikes from high-energetic particles like neutrons that can result in bit flips—is of increasing importance. The reason for these transient faults is that information is stored using far fewer atoms and electrons compared to past technology nodes.

Aging effects

Yet another problem during operation is that electrical properties change over time, a phenomenon known as *aging effects*. Various stress situations like thermal cycling trigger these effects. Such stress situations might cause the circuits to fail after a certain period of operation or to degrade or change its initial electrical and physical properties.

At a higher level of design abstraction, such effects could lead to new critical paths in the design. To date, we assumed that we can identify a critical path during design time and set up the system accordingly before it's put into operation (setting the operation frequency). However, this doesn't work any longer because these effects, even though present in conventional integrated circuits, speed up and amplify as we migrate to upcoming technology nodes.

All these effects will increase, and addressing them at the physical or device level isn't possible. In fact, we need a paradigm shift to build dependable systems with undependable devices (S. Borkar et al., "Microprocessors in the Era of Terascale Integration," *Proc. Design Automation and Test in Europe Conf.*, IEEE/ACM, 2007, pp. 237-242).

Assuming that we don't want to stop micro-miniaturization and want to keep Moore's law going, we must live with these effects. However, we probably must rethink everything in embedded computing, from the electronic design automation process and architecture to the software, middleware, and operating system. As the power consumption problem changed embedded computing through multicore architectures—which are far more power efficient than single-core systems clocked with higher frequencies—the dependability challenge will change embedded computing as well.

From now on, we must consider dependability a major design constraint and not handle it as an afterthought. Embedded systems will need a far higher degree of self-adaption and self-optimization to cope with these hardly predictable problems. We should make fewer design decisions at design time. At runtime, embedded systems need to self-detect faulty or degrading devices and self-adapt to ensure operability. We should address the dependability challenge at all levels of abstraction, including hardware and software design and architecture, operating system, and middleware.

SECURITY

Security provides significant challenges in networked embedded systems, especially as feature size decreases. The development of low-power, efficient security engines is a challenge, especially when considering the single-event upsets and the property of conventional cryptographic mechanisms that are very sensitive to all levels of noise. The development of effective mechanisms is especially important, considering the lack of security evaluation and assessment methodologies. Risk analysis and management are always a challenge as we discover or develop more weaknesses and attack methods for embedded systems.

However, viewing networked embedded systems as layered systems in which dependability and security are at the same layer lets us assess dependability and security together with the same or similar methodologies. We base our approach for dependability and security assessment and evaluation on the observation that security flaws are problems that are exploited on purpose. If someone exploited such a flaw by accident, it would be a dependability problem.

Thus, we can consider that failures and attacks differ only from the point of view of the motivation, since a weakness identified by chance can be exploited on purpose, and vice versa. In this fashion,

someone can exploit methods for dependability analysis for security risk assessment as well, at least at several levels of abstraction. Such methods can cover a significant void in assessing security in embedded and nonembedded systems. Thus, security assessment methods will clearly benefit from the body of work and methodologies of dependability analysis, considering the state of the art in both technical fields.

This unified approach to dependability and security assessment and evaluation will let embedded systems architects and designers face the challenges posed by the shrinking feature size and the increasing rates of errors at the design or runtime level.

Importantly, addressing the challenges originating from the changing technologies and security threads requires large-scale, multidisciplinary research efforts. This research will lead to significant changes in embedded computing.

Dimitrios Serpanos *is an associate professor in electrical and computer engineering at the University of Patras and ISI, Greece. Contact him at serpanos@ece.upatras.gr.*

Jörg Henkel *is a professor and chair of embedded systems at the University of Karlsruhe, Germany. Contact him at henkel@informatik.uni-karlsruhe.de.*

Editor: Wayne Wolf, School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta; wayne.wolf@ece.gatech.edu

Web 3.0: Chicken Farms on the Semantic Web

Jim Hendler, Rensselaer Polytechnic Institute

Emerging applications exploit the power of a new breed of semantic technologies.

The explosive growth of blogs, wikis, social networking sites, and other online communities has transformed the Web in recent years. The mainstream media has taken notice of the so-called Web 2.0 revolution—stories abound about events such as Facebook's huge valuation and trends like the growing Hulu-YouTube rivalry and Flickr's role in the current digital camera sales boom.

However, a new set of technologies is emerging in the background, and even the Web 2.0 crowd is starting to take notice.

THE SEMANTIC EDGE

One of the best-attended sessions at the 2007 Web 2.0 Summit (www.web2summit.com) was called "The Semantic Edge." Its theme was the use of semantic technologies to bring new functionality to such Web staples as search, social networking, and multimedia file sharing.

The session included beta demos by Metaweb Technologies (www.metaweb.com), which bills itself as "an open, shared database of the world's knowledge"; Powerset (www.powerset.com), a company building intelligent search tools with natural-language technology; and Radar Networks (www.radarnetworks.com), whose Twine tool, shown in Figure 1, aims to "leverage and contribute to the collective intelligence of your friends, colleagues, groups and teams."

Not included in the panel but working in the same space are other newcomers such as Garlik (www.garlik.com), a UK company creating tools to control personal information on the Web; online TV provider Joost (www.joost.com); Talis (www.talis.com), a vendor of software that makes data "available to share, remix and reuse"; and TopQuadrant (www.topquadrant.com), which offers consulting, teaching, and tool development in this space.

More established companies exploring semantic technologies for the Web include Mondeca (www.mondeca.com), a European enterprise information integration company, and Ontoprise (www.ontoprise.de), a German vendor of ontology-related tools. Big industry players like Oracle, Microsoft, and IBM are also getting into the game.

THE WEB AND WEB 2.0

All of this activity suggests that a new set of Web technologies is transitioning from toys and demos to tools and applications. Of course, this isn't the first time this has happened.

In the mid-1990s, the Web seemed to bloom overnight: Companies started putting Web addresses on their products, personal home pages began springing up, and Mark Andreesen's Mosaic browser got millions of downloads as more people discovered the World Wide Web.

The technology had actually been around for some time—Tim Berners-Lee created the Web in 1989—but it wasn't until this later time that it turned a knee in the growth curve and became one of the most important applications in history.

Another wave of technologies, dubbed Web 2.0 by Tim O'Reilly, began to emerge a few years later. Newspapers began losing subscribers to news blogs, encyclopedia companies woke up to discover Wikipedia was forcing them to change the way they work, and "google" became a verb on everybody's lips. Even those who weren't computer geeks began to talk about Flickr, YouTube, and Facebook.

Again, these technologies required time to mature, catch on virally, and turn that knee in the curve before they enjoyed widespread adoption.

TOWARD WEB 3.0

A new generation of Web applications, which technology journalist John Markoff called "Web 3.0" ("Entrepreneurs See a Web Guided by Common Sense," *The New York Times*, 12 Nov. 2006), is now starting to come to the public's attention. Companies like those showcased at the Web 2.0 Summit's "Semantic Edge" session are exploiting years of behind-the-scenes development, and there is growing excitement in the commercialization of what, until now, has been a slowly expanding wave of activity.

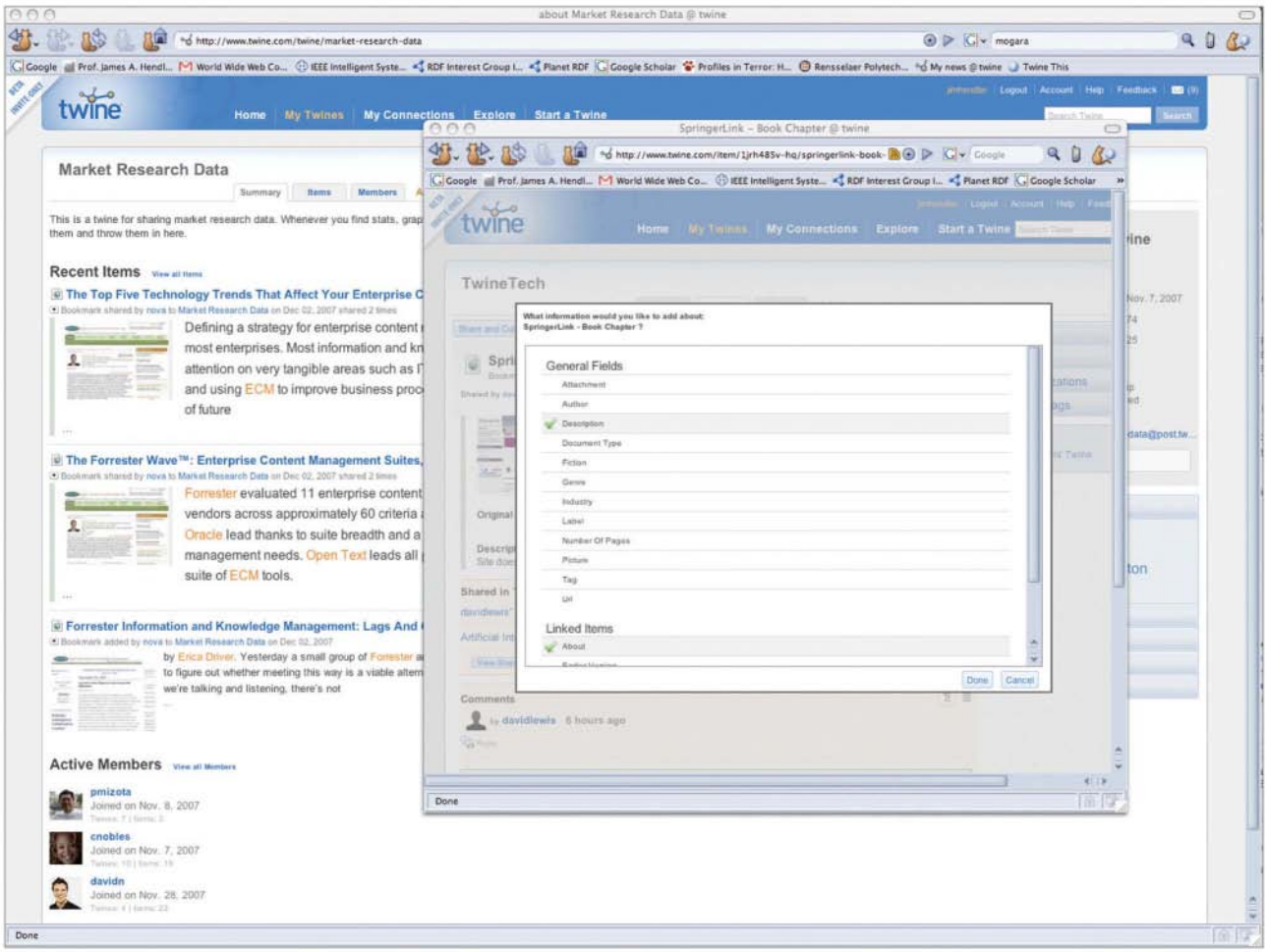


Figure 1. Twine, a beta tool released by Web 3.0 start-up Radar Networks, uses Semantic Web technologies to help users organize, find, and share online information.

Although semantic technologies have been around for a while, activity under the name "Semantic Web" really began to take off around 2000. Development of the Resource Description Framework was under way at the World Wide Web Consortium, which produced a first specification in 1999. However, the W3C metadata activity that had spawned it was inactive, and some original RDF supporters were shifting investment to other areas, such as XML and Web services, making it hard for the RDF adherents to find resources for further development.

The change came with an investment in the technology by the US Defense Advanced Research Projects Agency, which saw extending RDF as a way to deal with numerous interoperability problems plaguing the US Department of Defense, particularly with respect to sharing information across organizational boundaries. DARPA joined with the European Union's Information Society Technologies project, interested in similar issues, to form an ad hoc research group to explore how to apply some ideas from the field of AI to meet these needs.

This research investment brought together a curious mixture of Web gurus looking to bring data to the Web, AI practitioners starting to appreciate the power that scaling small amounts of semantics to Web size could provide, and visionary government data providers with interoperability problems that increasingly demanded solutions. These funds also supported development of early Semantic Web demos and tools that came to the attention of industrial researchers.

In 2001, the W3C renewed work in this area under the banner of the Semantic Web Activity (www.w3.org/2001/sw), and within a couple of years, new working groups were looking at improving the RDF standard; completing the standardization of RDF Schema (RDFS), a vocabulary definition language on top of RDF; and beginning work on OWL, an ontology language for the Web. In February 2004, new versions of RDF and RDFS, and the first version of OWL, became W3C Recommendations—standards for the Web.

CHICKEN-AND-EGG PROBLEMS

With any new technology, the transition from research to practice

and from standards to deployment imposes a time delay. This delay can sometimes be quite long, as a real chicken-and-egg problem arises: Tool vendors and manufacturers are reluctant to implement products until they see a market forming, but the market doesn't tend to form until the tools are available. The length of the delay thus typically depends on how soon vendors hear the demand from users and can get prototypes and tools to them.

However, the Semantic Web involves several other chicken-and-egg problems.

First, these applications require, in part or whole, data that is available for sharing either within or across an enterprise. Represented in RDF, this data can be generated from a standard database, mined from existing Web sources, or produced as markup of document content.

Machine-readable vocabularies for describing these data sets or documents are likewise required. The core of many Semantic Web applications is an ontology, a machine-readable domain description, defined in RDFS or OWL. These vocabularies can range from a simple "thesaurus of terms" to an elaborate expression of the complex relationships among the terms or rule sets for recognizing patterns within the data.

(While the Semantic Web community has long recognized that these different vocabulary levels fill different niches in the Web ecology, some critics mistakenly assume all Web ontologies are of the latter type. Overcoming this misunderstanding continues to be a challenge to the community.)

Finally, Web 3.0 applications require extensions to browsers, or other Web tools, enhanced by Semantic Web data. As in the early days of the Web when we were creating HTML pages without being quite sure what to do with them, for a long time people have been creating and exchanging Semantic Web documents and data sets without knowing exactly how Web applications would access and use them.

The advent of RDF query languages, particularly SPARQL (currently a W3C Candidate Recommendation), made it possible to create three-tiered Semantic Web applications similar to standard Web applications. These in turn can present Semantic Web data in a usable form to end users or to other applications, eliciting more obvious value from the emerging Web of data and documents.

However, motivating companies or governments to release data, ontology designers to build and share domain descriptions, and Web application developers to explore Semantic-Web-based applications all hinge on one another. Accomplishing this has sometimes been a daunting proposition.

RECENT TRENDS

Despite these challenges, the pace of semantic technology development has accelerated recently. In the early days of the technology, small companies tried—sometimes unsuccessfully—to create Semantic Web tools. During the past couple of years, however, larger companies have begun providing tools and technologies, both in product sets and open source offerings, and some of the biggest names in the data and software sectors have been testing the water.

Government data sets are being shared, small Semantic Web domain descriptions like the Friend of a Friend ontology are seeing great uptake (FOAF files currently number in the tens of millions), and SPARQL end points have motivated many Web application developers to seriously look at this technology. This in turn has led new start-ups to focus less on the tool market and more on user-facing applications.

Emerging Web 3.0 companies are combining the Web data resources, standard languages, ever-better tools, and (mostly simple) ontologies into applications that take advantage of the power of this new breed of semantic technologies. The entrepreneurs behind these efforts are exploiting the convergence of Semantic Web capabilities to embed small amounts of reasoning into large-scale Web applications, with tremendous potential.

It's an exciting time for those of us who have been evangelists, early adopters, and language designers for Semantic Web technology. What we see in Web 3.0 is the Semantic Web community moving from arguing over chickens and eggs to creating its first real chicken farms. The technology might not yet be mature, but we've come a long way, and the progress promises to continue for a long time to come.

Jim Hendler is the Tetherless World Senior Constellation Professor at Rensselaer Polytechnic Institute. Contact him at hendler@cs.rpi.edu.

Editor: Simon S.Y. Shim, CTO, MarkAny Inc., Seoul, Korea; sshim@markany.com.

Get the building blocks you need.

Take your career to the next level in software development, systems design, and engineering with:

- Article collections from the IEEE Computer Society
- Materials from Harvard Business School Publishing
- Computer discounts
- Online courses and certifications

Our experts. Your future.

www.computer.org/buildyourcareer

Continued from page 112

> The invention of the mechanical clock was one of a number of major advances that turned Europe from a weak, peripheral, highly vulnerable outpost of Mediterranean civilization into a hegemonic aggressor. Time measurement was at once a sign of new-found creativity and an agent and catalyst in the use of knowledge for wealth and power. (Davis S. Landes, *Revolution in Time*, Harvard University Press, 1983, p. 12.)

This knowledge came from measurement and reckoning, and it still does. Scientists develop the knowledge, engineers apply it. What has changed is knowledge about how to measure and reckon. Until less than 50 years ago, the slide rule was to the engineer what the stethoscope is to the doctor.

Accounting

Scientists and engineers do not directly produce wealth and power in society. They merely provide the means for some of society to accumulate property through management and trade. The owners also use numbers, but not in the same way scientists and engineers do.

Property comes in a variety of forms, and the numbers used in its management—counts, dates, and prices—cling to names identifying and describing the property. Early forms of accounting 10 millennia ago in southwest Asia used clay tokens whose shape identified the property and whose number denoted the quantity (www.utexas.edu/cola/centers/lrc/numerals/dsb/dsb1.html). Later, these tokens were embedded in clay bullae that served as delivery dockets, and later still tokens were pressed into the surface of a bulla before baking so that its content could be known without breaking it open (Steven Roger Fischer, *A History of Writing*, Reaktion Books, 2001). It would seem likely that, starting around five millennia ago, this led to the cuneiform tablets used primarily for accounting.

Although many cultures developed other forms of recording in accounts and dockets, double-entry bookkeeping provided a significant formalization. Started in Italy around the 15th century, it provided a simple means of validating accounts. The arithmetic was troublesome for accountants, not because it was complex—rarely extending beyond addition and subtraction—but because many numbers were involved. The owners themselves usually employed clerks to do the transcription and arithmetic.

The more direct use of writing for social control is arguably why its technology advanced sooner and faster.

A century ago, governments and businesses adopted punched cards for recording census and accounting data. Machinery to process such data soon followed. This type of accounting, called unit record, maintained each unit record as a separate physical entity, which allowed files to be sorted, merged, split, and printed from without having to copy them. Keypunch and verifier operators transcribed data from documents to cards, and teams of operators moved files of cards from machine to machine—files of thousands, even millions of cards. Experienced operators put together the operational procedures and plugged program panels.

Writing

Most cultures do not spread wealth and power uniformly through society. Hierarchical by nature, most societies concentrate wealth and power at the top and attenuate it all the way down. Communication maintains the hierarchy, which in preliterate societies was limited in time and in distance, with personal memory being used to preserve ideas and commands over time, and drum or whistle languages over distance.

Physical representation overcomes both limitations at once. The clay bullae and tablets of five millennia past combined impressed numbers and pictographic names, and the pictographic writing evolved into the hieroglyphics of Egypt that some scholars argue triggered the Chinese writing method. The hieroglyphs of Egypt were written by scribes and used by priests, and mandarins wrote and used the logograms of China. The hieroglyphs of Egypt evolved into widely used alphabetic systems of writing more directly based on lower-level speech components, which made them more effective for the administrative control and propaganda needed to sustain and expand hierarchical societies.

Perhaps because writing provided a more significant tool for social control than reckoning or accounting, its technology received more attention. Scribes used papyrus, vellum, parchment, and other bases for writing on as individual message sheets and scrolls and later as bound books. The expense of keeping scribes to transcribe important books, often imperfectly, led to the development of printing technology, first with the hand press, starting about five centuries ago, then with the machine press about two centuries ago (Philip Gaskell, *A New Introduction to Bibliography*, Oak Knoll Press, 2000).

In the early stages of printing, the industry replaced scribes with skilled workers like typecasters, compositors, pressmen, and binders—and with professionals like master printers and punch cutters. The later stages of printing involved different machines and more trades, as well as professionals like authors and illustrators.

INTERLUDE

This early history shows that writing developed from accounting, which had developed from reckoning. The more direct use of writing for social control is arguably why its technology advanced sooner and faster than the technology of accounting and reckoning.

Increasing cheapness of printing led eventually to wider literacy and a greater variety of publications. Cheaper printing also led to books being used outside religious and other administrative circles and to mass production of impermanent products like pamphlets and newspapers. Widespread use of printing by press, duplicator, copier, and personal printer or typewriter led to the characteristic printing trades and professions becoming subsumed in the activities for which the printing was used. Such craftsmen, while still there, were pushed into the background.

The development of modern computing was like the earlier development of printing and publishing, but it focused on reckoning and accounting. It might well be possible to predict the computing profession's future from what happened to the people dependent on printing. One difference is that both reckoners and accountants adopted the electronic computer at about the same time.

THE ELECTRONIC COMPUTER

Reckoners, the accumulators of knowledge, needed automatic computing machinery because their modeling of accumulated complexity and manual simulation became too slow and fallible. Analog computers helped for a while, but they lacked the required accuracy. Accountants, the managers of property, needed similar machinery because the traditional punched card methods were slow and labor-intensive. Arithmetic was not the problem, expense was.

Early scientific computers typically used binary arithmetic, and they were used for "number crunching," with the results stored on paper and magnetic tape. Early commercial computers used decimal arithmetic for automatic data processing, spawning common initialisms such as ADP, EDP, and, simply, DP. Data arrived on punched cards and left as output to a high-speed printer. Master files resided for a while on magnetic tape, although the soon-to-be-developed magnetic discs would allow direct access to within those master files, particularly for inquiry.

Reckoners operated their own machines and wrote their own programs. Because their computers were expensive, time-sharing systems were soon developed so that users could reckon on their computer simultaneously. Because programming required some special skills, accountants hired programmers who worked apart from the operators. Management promoted experienced programmers to become system analysts, who specified needed programs rather than coding them.

Cheap personal computers and networking changed the world dramatically.

Even though increased production led to computers with both binary and decimal arithmetic, scientific and commercial computing remained distinct. Technical computing usually involved scientists and engineers working closely with their computing departments. In the commercial world, the DP department became a highly political entity within an organization, an entity that fought for power by undertaking only large projects and dictating to end users the capabilities they were to have.

Because their prospective users built the first electronic computers, courses in computing at universities usually started in technical departments such as physics or electrical engineering—and stayed there. As commercial computing evolved, commerce and management schools realized their students needed training in computing and set up computing departments with their redundancy hidden under names like Information Systems.

The dichotomy also appeared in professional computing organizations. In countries with a single computer society, control of the society typically oscillated between the scientists and technologists, who saw themselves as naturally in charge of the profession, and the commercial practitioners, who saw the academics as out of touch with reality.

The availability of cheap personal computers and networking has changed the world dramatically, and not just within the computing profession. Machines like typewriters and vocations like typing and stenography have almost disappeared. DP (now IT) departments still exist in large organizations and still focus on large projects that typically fail to meet their original objectives, but small organizations now get along without them.

However, professional computing courses and organizations are withering. With cheap computers and networking available, people see them as part of everyday life, both at work and in the home. What need is there, then, for computing professionals?

The education and computing sectors could work together to reconfigure the way computing is used and to provide more benefit to society from digital technology. This would also reconfigure the computing profession and give it a more productive future. My essay of last January outlines one possible approach. But whatever approach we take, the computing profession must act soon and vigorously to avoid the old printing profession's fate.

***Neville Holmes** is an honorary research associate at the University of Tasmania's School of Computing and Information Systems. Contact him at neville.holmes@utas.edu.au.* *In addition to the books cited here, he would recommend Karl Menninger's classic* Number Words and Number Symbols *(MIT Press, 1969).*

Editor: Neville Holmes, School of Computing and Information Systems, University of Tasmania; neville.holmes@utas.edu.au.

The History of the Computing Profession

Neville Holmes, University of Tasmania

With cheap computers and networking available, who needs computing professionals?

My January 2007 essay ("The Computing Profession and Higher Education," pp. 116, 114-115) prompted an invitation for me to lead a discussion about the computing profession at an IEEE/IEAust evening meeting in Brisbane. Asked also to speak at a lunchtime meeting at the University of Queensland, I put together a presentation titled "The Early Development of the Computing Profession" designed to attract students to the evening meeting.

In one sense, the lunchtime gathering proved very successful, filling the meeting room, prompting attendees to ask many questions and to have me go on long past the lunch hour's end. In another sense, however, my presentation failed completely because no students attended the evening meeting. The reason became clear in hindsight: I had overloaded the presentation with URLs for early machinery, mainly machinery I had used and could explain to questioners (see http://eprints.utas.edu.au/1301). Thus, I spent most of the time explaining what used to be done and relatively little time considering what kind of people did that work.

The pity of this is that the early computing people were far more interesting than the machinery. Further, the people I did talk about made their contributions relatively recently. With this essay, I seek to redress the balance and, more importantly, to provoke discussion of the profession's future.

EARLY HISTORY

Computation, according to the usual dictionary definitions, addresses the manipulation of numeric values. My *Oxford English Dictionary* defines *computing* primarily as the "act of calculation or counting." The term has developed a much wider meaning today, with few digital computer applications focusing on arithmetic. This has occurred because different formal social activities developed with distinct human vocations, all of which modern digital technology supports at various stages and to various degrees.

Human society is based on language, the first digital technology, one that started long ago and originally involved only speech and gesture. The use of tools in digital technology developed relatively early in three areas: reckoning and mathematics, accounting and bookkeeping, and writing and communication.

Reckoning

Numbers formed an early part of human culture and language, as suggested by their rich use in various forms in preliterate societies (Marcia Ascher, *Ethnomathematics*, Wadsworth, 1991).

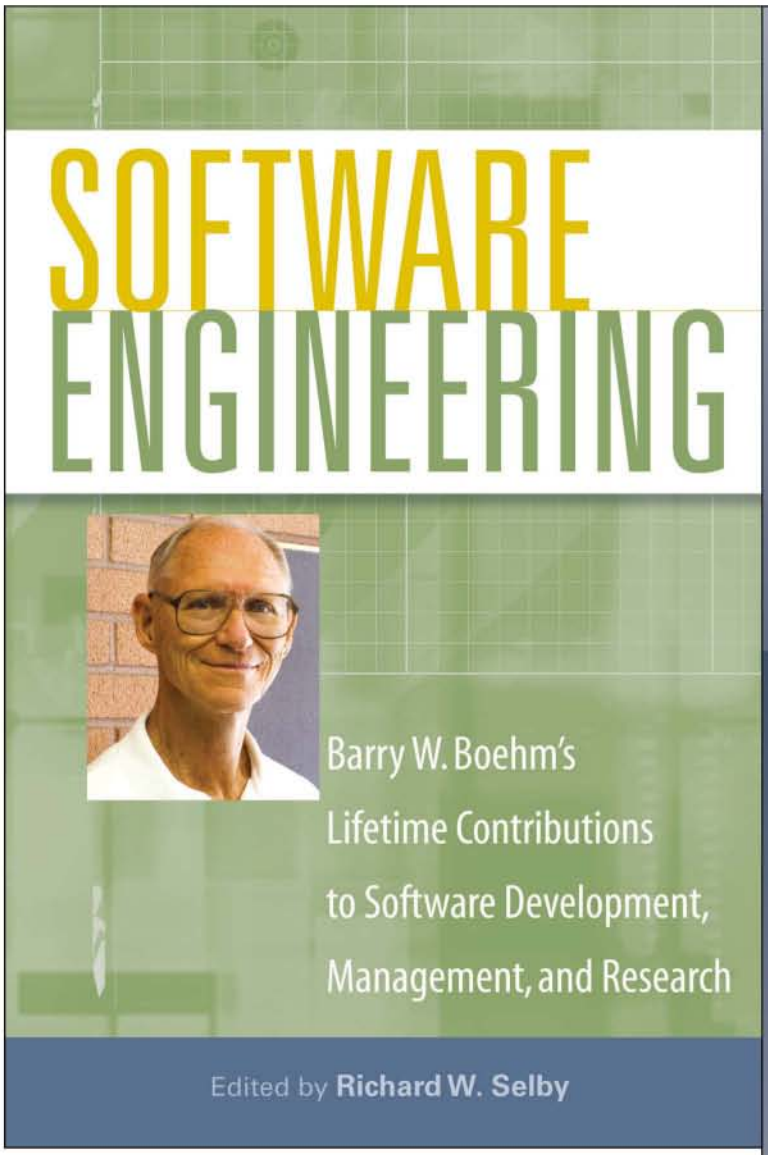
The most significant use of numbers was and remains in understanding how things work. Predicting the moon's phases once played an important role in scheduling nocturnal activities. An eagle bone discovered in France and measured to be 13,000 years old is notched in a manner strongly suggesting a recording of the lunar month's days (David Ewing Duncan, *The Calendar*, Fourth Estate, 1998). Other European artifacts, two or three times older, show similar if less strongly suggestive markings.

People required more formal calendars when agriculture developed. Many of these, based on lunar months, were numerically complex. Almost six millennia ago, the Egyptians set up a solar calendar of 365 days that they used for four millennia. Although halfway through this period astronomers calculated the need for a leap day every fourth year, the priestly bureaucracy prevented this reform.

The socially significant time of day is solar. Sundials are simple and useful, providing the sun is out, but people also developed other devices for measuring time. The most significant development—the escapement clock—provided perhaps the first analog-to-digital conversion device. A Chinese astronomical clock used an hydraulic escapement a thousand years ago, but the European adoption three centuries later of an oscillatory drive for the escapement was highly significant:

Continued on page 110

FEATURED TITLE FROM WILEY AND CS PRESS



Software Engineering:
Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research

edited by
Richard W. Selby

978-0-470-14873-0
June 2007 • 832 pages
Hardcover • $79.95
A Wiley-IEEE CS Press Publication

To Order:
North America
1-877-762-2974
Rest of the World
+ 44 (0) 1243 843294

This is the most authoritative archive of Barry Boehm's contributions to software engineering. Featuring 42 reprinted articles, along with an introduction and chapter summaries to provide context, it serves as a "how-to" reference manual for software engineering best practices. It provides convenient access to Boehm's landmark work on product development and management processes. The book concludes with an insightful look to the future by Dr. Boehm.

20% Promotion Code CSCH7

WILEY Publishers Since 1807

IEEE

IEEE computer society

CALL FOR PARTICIPATION

IEEE VISUALIZATION 08 • IEEE INFORMATION VISUALIZATION 08

OCTOBER 19 - 24, 2008 • COLUMBUS, OHIO, USA

Vis 2008 and **InfoVis 2008** are the premier forums for data and information visualization advances for academia, government, and industry. These events bring together researchers and practitioners with a shared interest in tools, techniques, and technology. The conferences will include an exciting and informative collection of workshops, tutorials, papers, panels, demonstrations, posters, and exhibitions. We invite you to participate by sharing your research, insights, experience, and enthusiasm in Columbus, Ohio.

Paper Abstracts Due: March 21, 2008

Full Papers Due: March 31, 2008

Check the web sites for the complete list of deadlines.

Vis Conference Chairs:
Raghu Machiraju, *The Ohio State University*
Roger Crawfis, *The Ohio State University*
Ken Joy, *University of California, Davis*

InfoVis Conference Chair:
Jarke van Wijk, *Eindhoven University of Technology*

VAST Symposium Chairs:
David Ebert, *Purdue University*
Thomas Ertl, *University of Stuttgart*

Co-located with Vis and InfoVis 2008 is the following symposium:

VAST 2008: IEEE Symposium on Visual Analytics Science and Technology

http://vis.computer.org/vis2008

http://conferences.computer.org/infovis/infovis2008

http://conferences.computer.org/vast/vast2008

For questions, email:
Vis: info@vis.computer.org
InfoVis: infovis@vis.computer.org
VAST: vast@vis.computer.org

Sponsored by the IEEE Computer Society
Visualization and Graphics Technical Committee.

IEEE vgtc IEEE IEEE COMPUTER SOCIETY