Innovative Technology for Computer Professionals

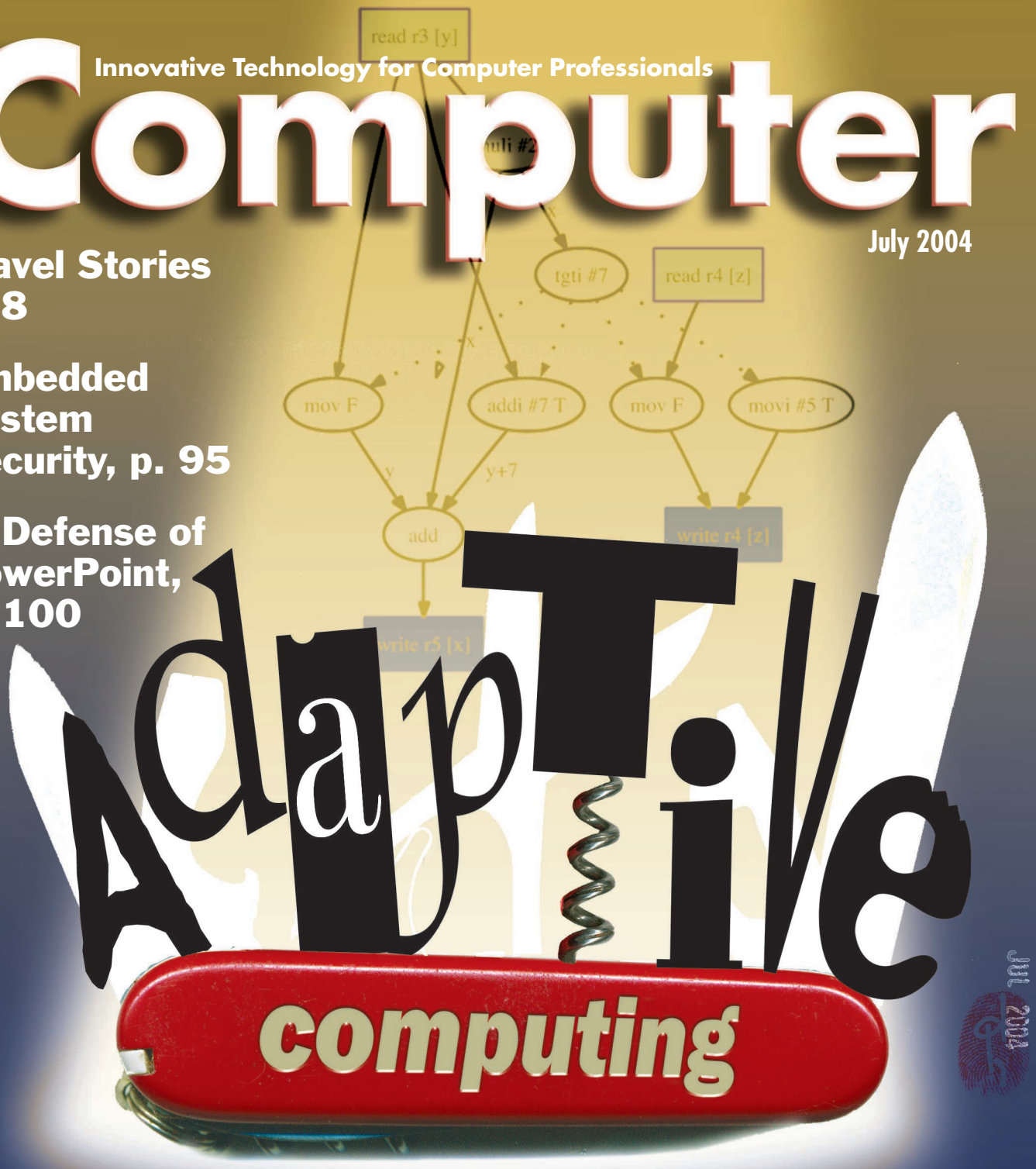
Computer

July 2004

Travel Stories
p. 8

Embedded System Security, p. 95

In Defense of PowerPoint, p. 100



IEEE

IEEE COMPUTER SOCIETY

http://www.computer.org

IPDPS 2005

Sponsored by the IEEE Computer Society Technical Committee on Parallel Processing

In cooperation with ACM SIGARCH, IEEE Computer Society Technical Committee on Computer Architecture, and IEEE Computer Society Technical Committee on Distributed Processing

Hosted by Colorado State University

19th IEEE International Parallel & Distributed Processing Symposium

Monday, 4 April – Friday, 8 April 2005
Omni Interlocken Hotel • Denver, Colorado

CALL FOR PARTICIPATION

IPDPS serves as a forum for engineers and scientists from around the world to present their latest research findings in the fields of parallel processing and distributed computing. The five-day program will follow the usual format of contributed papers, invited speakers, panels, industrial track, and exhibits mid week, framed by workshops held on Monday and Friday. During the week, participants will have an opportunity to organize Birds-of-a-Feather (BOF) sessions, and a special tutorial will be offered. Program details will be posted on the Web, so you are encouraged to regularly check the IPDPS Web site at www.ipdps.org for updates. General email inquiries should be addressed to info@ipdps.org.

CALL FOR PAPERS

Authors are invited to submit manuscripts that demonstrate original unpublished research in all areas of parallel and distributed processing, including the development of experimental or commercial systems. Topics of interest include but are not limited to:

1. Parallel and distributed algorithms, including stability and fault tolerance of distributed systems, communication and synchronization protocols, network algorithms, and scheduling and load balancing.

2. Applications of parallel and distributed computing, including web applications, peer-to-peer computing, grid computing, scientific applications, and mobile computing.

3. Parallel and distributed architectures, including shared memory, distributed memory (including petascale system designs, and architectures with instruction-level and thread-level parallelism), special-purpose models (including signal and image processors, network processors, other special purpose processors), nontraditional processor technologies, network and interconnect architecture, parallel I/O and storage systems, system design issues for low power, design for high reliability, and performance modeling and evaluation.

4. Parallel and distributed software, including parallel programming languages and compilers, operating systems, resource management, middleware, libraries, data mining, and programming environments and tools.

BEST PAPER AWARDS

Awards will be given for one best paper in each of the four conference technical tracks: algorithms, applications, architectures, and software. The selected papers will also be considered for possible publication in a special issue of the Journal of Parallel and Distributed Computing.

WHAT TO SUBMIT

Submitted manuscripts may not exceed 16 single-spaced pages using 12-point size type on 8.5x11 inch pages, including figures and tables. References may be included in addition to the 16 pages. Submissions will be judged on correctness, originality, technical strength, significance, quality of presentation, and interest and relevance to the conference attendees. Submitted papers may not have appeared in or be under consideration for another conference or a journal. Submission procedures are available via Web access at www.ipdps.org. For those who have only e-mail access, send an e-mail message to cfp@ipdps.org for an automatic reply that will contain detailed instructions for submission of manuscripts. If no electronic access is available, contact the Program Chair at the address given below.

All manuscripts will be reviewed. Manuscripts must be received by **October 8, 2004**. Notification of review decisions will be mailed by December 17, 2004. Camera-ready papers will be due January 21, 2005. IPDPS 2005 Proceedings for both contributed papers and workshops will be published by the IEEE Computer Society Press on CD-ROM and will be distributed at the Symposium along with a hard copy volume of abstracts.

www.ipdps.org

colorado

GENERAL CO-CHAIRS

H.J. Siegel, Colorado State University, USA
David A. Bader, University of New Mexico, USA

GENERAL VICE CHAIR

Charles Weems, University of Massachusetts at Amherst, USA

PROGRAM CHAIR

Jean-Luc Gaudiot
gaudiot@uci.edu
Department of Electrical Engineering and Computer Science
University of California, Irvine, CA 92697-2625 - USA

PROGRAM VICE-CHAIRS

ALGORITHMS
Albert Y. Zomaya, University of Sydney, Australia

APPLICATIONS
Yves Robert, École Normale Supérieure de Lyon, France

ARCHITECTURES
Kemal Ebcioğlu, IBM T.J. Watson Research Center, USA

SOFTWARE
Dan Reed, University of North Carolina at Chapel Hill, USA

IEEE is FOR PEOPLE. NOT PROFIT.

IEEE publishes information for the advancement of science — not for profit. As an IEEE member, Sanjay wouldn't have it any other way.

Sanjay wants his research to improve lives around the world. He relies on IEEE for the latest science and technology standards. And thanks to IEEE editors and peer reviewers, Sanjay can trust the information he finds.

Every month, millions of researchers depend on IEEE for the latest results in their fields. With 120 journals and magazines, 350 annual conferences and more than 900 active standards, IEEE publishes the science that helps make the world a better place.

To Sanjay, IEEE is for people, not profit.
Discover what IEEE can be for you.

Go here.

www.ieee.org/discover

IEEE

IEEE INFORMATION DRIVING INNOVATION

Seapine Software™
changing the world of software development

Developers love creating code... Managers crave process control... Bridge the gap with Seapine CM.

Software development is a team effort with developers, testers, and management all working toward one goal – delivering the highest quality product on time.

Built on award-winning TestTrack Pro and Surround SCM, Seapine CM brings structure to source control and issue management, improving communication while accelerating product development.

Seapine CM helps your team...

Define custom change request workflows, putting you in control of who makes changes and who authorizes the closure of issues.

Associate source code changes with defects or change requests.

Gain a thorough understanding of how much work remains before project completion.

View complete audit trails of what changed, why, and by whom.

Understand how close you are to release—how many issues are open, how quickly are you closing them, how many are re-opened?

Successful team-based development requires the proper process supported by the right development tools. Tools that are flexible, easy to use, secure, and scalable—like Seapine CM.

Features:

Complete source code control with private workspaces, automatic merging, role-based security and more.

Comprehensive defect management—track change requests, bug reports, feature requests and more.

Fast and secure remote access to your source files and defects—work from anywhere.

IDE integration with JBuilder, Visual Studio, Dreamweaver, and other leading development tools.

Advanced branching, triggers, email notifications, fully configurable workflows, and customizable fields put you in complete control of your process.

Open interfaces—triggers, email notifications, SOAP support, XML data exchange, and ODBC clients.

Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

Achieve major improvements in software development performance through better tool integration and process automation. Manage defects, development issues, and change requests with award-winning TestTrack Pro and gain complete control over your source code and change process with Surround SCM. Seapine's integrated change management tools are feature rich, highly scalable, Web enabled, and cross platform. Streamline your development process with Seapine CM and help your team deliver quality software products on time, every time.

Learn more about the Seapine CM suite at www.seapine.com or call 1-888-683-6456

Surround SCM Source Control

TestTrack Pro Defect Management

QA Wizard Automated Testing

software development 13th annual product excellence award

Innovative Technology for Computer Professionals

Computer

July 2004, Volume 37, Number 7

Cover design and artwork by Dirk Hagner

ABOUT THIS ISSUE

Given the increasing complexity of information and computer systems, the industry is beginning to explore adaptive software systems that are self-configuring, self-optimizing, and self-repairing. At the same time, hardware designers are exploring systems that dynamically reconfigure resources to optimize performance for different types of applications. In this issue, we look at a new, post-RISC class of instruction set architectures intended to match semiconductor technology evolution over the next decade, technologies for composing adaptable software, and a thick-client approach to rapid personalization of anonymous hardware for transient use.

PERSPECTIVES

27 **Who Is Liable for Insecure Systems?**
Nancy R. Mead
A survey of security-related liability issues shows that as data crime increases and the costs of hacking and other malicious behavior soar, vendors must make their software more secure or suffer market and legal consequences.

COMPUTING PRACTICES

36 **Issues in High-Speed Internet Security**
Peder Jungck and Simon S.Y. Shim
Protecting networks against fast-moving threats such as the SQL Slammer worm requires a new paradigm that offers flexibility, high performance, and speed.

COVER FEATURES

44 **Scaling to the End of Silicon with EDGE Architectures**
Doug Burger, Stephen W. Keckler, Kathryn S. McKinley, Mike Dahlin, Lizy K. John, Calvin Lin, Charles R. Moore, James Burrill, Robert G. McDonald, William Yoder, and the TRIPS Team
The TRIPS architecture is the first instantiation of an EDGE instruction set, a new, post-RISC class of instruction set architectures intended to match semiconductor technology evolution over the next decade, scaling to new levels of power efficiency and high performance.

56 **Composing Adaptive Software**
Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H.C. Cheng
Compositional adaptation enables software to modify its structure and behavior dynamically in response to changes in its execution environment. A review of current technology compares how, when, and where recomposition occurs.

65 **Seamless Mobile Computing on Fixed Infrastructure**
Michael Kozuch, Mahadev Satyanarayanan, Thomas Bressoud, Casey Helfrich, and Shafeeq Sinnamohideen
Internet Suspend/Resume is a thick-client approach to mobility in which hardware virtualization and file caching are the keys to rapid personalization of anonymous hardware for transient use.

RESEARCH FEATURES

73 **Policy-Based Dynamic Reconfiguration of Mobile-Code Applications**
Rebecca Montanari, Emil Lupu, and Cesare Stefanelli
A policy-based implementation framework supports high-level reconfiguration strategies that separate mobility concerns from application functionality.

Computer

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

OPINION

8 **At Random**
Travel Stories
Bob Colwell

NEWS

13 **Industry Trends**
Open Source Databases Move into the Marketplace
Linda Dailey Paulson

20 **Technology News**
Is MIMO the Future of Wireless Communications?
George Lawton

24 **News Briefs**
The New Dope on Semiconductor Doping ■ PARC Develops Software to Connect Devices ■ System Uses Existing Attacks to Predict Future Threats

MEMBERSHIP NEWS

81 **Call and Calendar**

84 **Computer Society Connection**

COLUMNS

92 **IT Systems Perspectives**
A Critical Look at Open Source
Brian Fitzgerald

95 **Embedded Computing**
Embedded System Security
Philip Koopman

100 **The Profession**
In Defense of PowerPoint
Neville Holmes

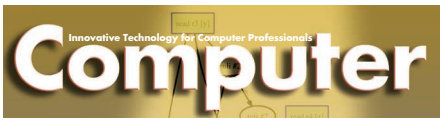
DEPARTMENTS

4 Article Summaries
6 Letters
11 32 & 16
12 IEEE Computer Society Membership Application
82 Bookshelf
83 Products
88 Career Opportunities
91 Advertiser/Product Index

NEXT MONTH:
Sensor Networks

Membership Magazine of the

IEEE COMPUTER SOCIETY



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 $17 for a subscription to *Computer* magazine. Nonmember subscription rate available upon request. Single-copy prices: members $20.00; nonmembers $88.00.

Postmaster: Send undelivered copies and address changes to *Computer*, IEEE Service Center, 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 4960-2 Walker Road, Windsor, ON N9A 6J3. Printed in USA.

ARTICLE SUMMARIES

Who Is Liable for Insecure Systems?

pp. 27-34

Nancy R. Mead

Over the past several years, we have seen both the enactment of legislation affecting liability and the appearance of actual liability cases in the courts.

Although several software flaws can make systems insecure, an entire industry including software vendors, systems administrators, consultants, network technicians, and clearing houses has developed to try to mitigate security holes after the fact.

Nevertheless, the issue of liability lurks just beneath the surface of all their activities. Although the courts may not be the best venue for resolving this, liability cases will take place and an evolution of best practices will occur as well.

Issues in High-Speed Internet Security

pp. 36-43

Peder Jungck and Simon S.Y. Shim

Over the past decade, the threat of computer worms and viruses has grown from a nuisance to perhaps the greatest obstacle to the Internet's growth and reliability. Trends in worm and virus delivery mechanisms and infection speed have also changed. Today, too often the first sign of a virus is that a part of the network goes down. Flash worms such as SQL Slammer have paved the way for future worms to carry payloads that directly target their victims and wreak havoc on government, business, and societal structures.

To address both the threats facing networks today and future scalability demands, we need new security methodologies, deployment strategies, systems, and architectures. New breeds of systems based on innovative processing components will help achieve flexible line rate high-speed security over time.

Scaling to the End of Silicon with EDGE Architectures

pp. 44-55

Doug Burger, Stephen W. Keckler, Kathryn S. McKinley, Mike Dahlin, Lizy K. John, Calvin Lin, Charles R. Moore, James Burrill, Robert G. McDonald, William Yoder, and the TRIPS Team

Post-RISC microprocessor designs must introduce new ISAs to address the challenges that modern CMOS technologies pose while also exploiting the massive levels of integration now possible. To meet these challenges, the TRIPS Team at the University of Texas at Austin has developed a new class of ISAs, called Explicit Data Graph Execution, that will match the characteristics of semiconductor technology over the next decade.

EDGE architectures appear to offer a progressively better solution as technology scales down to the end of silicon, with each generation providing a richer spatial substrate at the expense of increased global communication delays.

Composing Adaptive Software

pp. 56-64

Philip K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H.C. Cheng

Compositional adaptation exchanges algorithmic or structural system components with others that improve a program's fit to its current environment. With this approach, an application can add new behaviors after deployment. Compositional adaptation also enables dynamic recomposition of the software during execution. While dynamic software recomposition dates back to the earliest days of computing, such programs were difficult to write and debug. Several new software tools and technologies now help address these problems.

The authors review the research in compositional adaptation and survey the supporting technologies, proposed solutions, and areas that require further study.

Seamless Mobile Computing on Fixed Infrastructure

pp. 65-72

Michael Kozuch, M. Satyanarayanan, Thomas Bressoud, Casey Helfrich, and Shafeeq Sinnamohideen

The authors envision a world in which computers are provided for public use in locations ranging from coffee shops to medical office waiting rooms. In such a world, only when a user starts to use a computer will it acquire his unique customization and state, which will likewise disappear when he stops using it.

For this to be a compelling vision from a user's viewpoint, the customization and state acquisition process must be accurate and nearly instantaneous. For it to be a viable business model, the management and system administration costs of pervasive deployments of machines must be low. To address these challenges, the authors have developed *Internet Suspend/Resume*, a pervasive computing technology that rapidly personalizes and depersonalizes anonymous hardware for transient use.

Policy-Based Dynamic Reconfiguration of Mobile-Code Applications

pp. 73-80

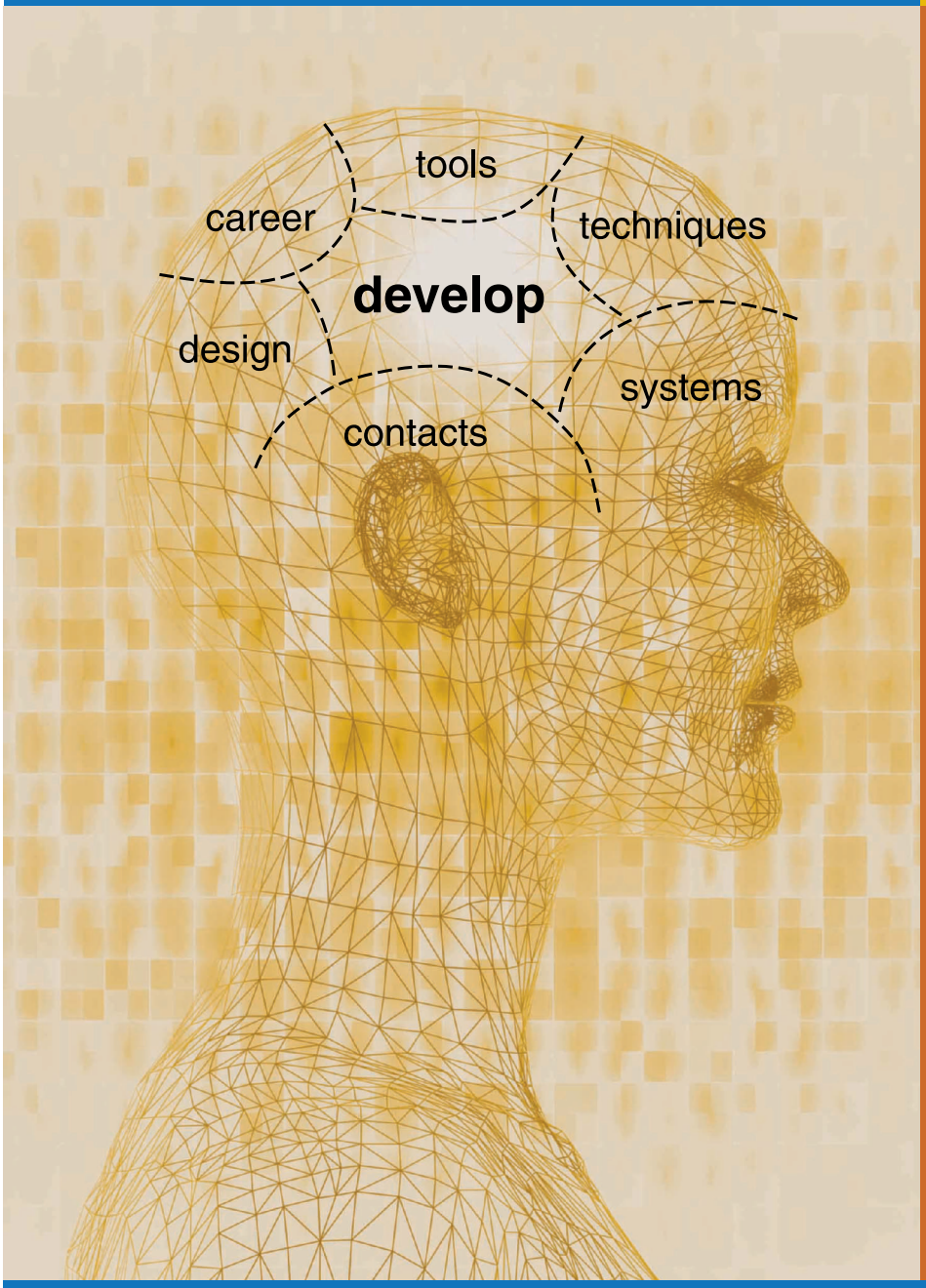
Rebecca Montanari, Emil Lupu, and Cesare Stefanelli

Code mobility enables dynamic customization and configuration of ubiquitous Internet applications. Mobile applications can transfer the execution of software components from one device to another depending on resource availability. They can also adapt functionality according to user needs and device characteristics. Thus, the authors have developed a policy-based approach to mobility programming that expresses and controls reconfiguration strategies at a high level of abstraction, separate from the application's functionality.

Embedded Systems Conference Boston

Co-located with: Embedded Security seminar

September 13-16, 2004
Hynes Convention Center
Boston, MA



Develop Your Career

Attend the East Coast's largest embedded systems design educational opportunity

The Embedded Systems Conference Boston

Keep current with 78 classes and full-day tutorials taught by more than 40 industry experts addressing your toughest processor-based design challenges. The event features a special focus on Single Board Computers including classes, a panel, and a Single Board Computer Village on the exhibits floor.

The Embedded Security Seminar

Learn to apply security to your connected systems with this three-day technical program offering sessions on topics including encryption, standards, firewalls, protecting IP, and security for wireless and Internet protocol.

Maximize Your Knowledge. Attend both conferences in the same week and save!
Design your own Combo Pass conference package choosing from 92 technical sessions from the **Embedded Systems Conference Boston** and the **Embedded Security Seminar.**

Save up to $815, register by July 13th.

www.embedded.com/esc

Priority code: ZJD

Conference Sponsors:

EETIMES EmbeddedSystems PROGRAMMING

CMP United Business Media

JAVA THREADS

In "Coping with Java Threads" (Bo Sandén, Apr. 2004, pp. 20-27), I am concerned that a recurring comment is wrong. As a result, the author recommends actions that reduce the robustness of Java applications. Specifically, he recommends against exclusion synchronization for problem-domain resources, instead simulating a semaphore via condition synchronization. This increases the chance of errors that never release the resource, leaving the resource unavailable to the rest of the application.

The author states that exclusion synchronization forces waiting threads to spin. This is incorrect. Exclusion synchronization puts requesting threads on the requested object's wait set. Three sources describe this behavior: the JVM specification, a JVM implementation, and a sample application.

The JVM spec (java.sun.com/docs/books/vmspec/2nd-edition/html/VMSpecTOC.doc.html) details the monitorenter and monitorexit opcodes, which implement synchronization.

The Java class below also demonstrates this behavior. It was executed on Windows and Linux by Sun's JVM. In both cases, the process was virtually always in the "sleep" state even though the "waiting" thread was trying to access the Synchronized method "neverReturn."

```
public class ThreadTest {
   public static class Synchronized {
      public synchronized void neverReturn(String name) {
         while (true) {
            try {
               Thread.sleep(10000);
               System.out.println(name);
            }
            catch (InterruptedException ex){
            }
         }}}
   public static class Sleepy implements Runnable {
      private Synchronized synched;
      private String name;
      public Sleepy(Synchronized synched, String name) {
         this.synched = synched;
         this.name = name;
      }
      public void run() {
         synched.neverReturn(name);
      }}
   public static void main(String[] args) {
      Synchronized synched = new Synchronized();
      Thread sleepy = new Thread(new Sleepy(synched, "sleepy"));
      sleepy.start();
      Thread waiting = new Thread(new Sleepy(synched, "WAITING"));
      waiting.start();
   }
}
```

Sample Java class implementing synchronization.

I respect the accomplishments of the Ada language and community. However, in my opinion, Ada has been overcome by events. Java has generated much research and commercial interest, and, as a result, it continues to improve and mature as a technology.

Jeff D. Sparkman
Huntsville, Ala.
jeff.beth.sparkman@ieee.org

The author responds:

Jeff Sparkman correctly observes that exclusion synchronization does not require threads to test a lock variable repeatedly, which I refer to as "spinning." Spinning is one implementation technique, but others, involving thread suspension/queuing, are also possible. So threads may indeed be waiting on an object lock as Mr. Sparkman suggests. I oversimplified this and should have been more precise.

The assertion that "exclusion synchronization puts requesting threads on the requested object's wait set" is incorrect, however. A thread cannot enter the wait set simply by calling a synchronized operation on a locked object or attempting to enter a synchronized block. The mechanism for exclusion synchronization is separate from the wait set mechanism. Thus, the wait set can be optimized for longer waits and multiple waiting threads, and the exclusion synchronization mechanism can be optimized for short waits and high performance.

I disagree with the statement that Ada has been "overtaken by events." I certainly don't expect it to be widely used for general-purpose programming. But it is still a living and evolving language, especially for safety-critical systems. A new version of the Ada standard is due in 2005, and "safe subsets" such as the Ravenscar tasking profile place Ada at the leading edge of

high-integrity technology.

Even with real-time enhancements, Java is still a largely untested and thus risky technology for real-time and safety-critical applications. As my article attempts to show, the subtleties of Java's thread model are easily misunderstood. Unfortunately, many misuse-prone constructs are inherent in the model and cannot readily be amended.

IT SYSTEMS MANAGEMENT

The authors of "Managing Systems Development" (G. Richardson and B. Ives, IT Systems Perspectives, Mar. 2004, pp. 93-94) made a good point when they stated that management should view IT project development as a business rather than as a technical activity.

More often than not, organizations incorrectly characterize IT services as a department that merely spends money. In some cases, management does not understand that in addition to increasing efficiency, technology improves the entire working environment.

Because their department is mission critical, IT professionals bear tremendous responsibilities, some of which can be quite nerve-racking, especially if they are systems administrators. One way to help the IT department get the resources it needs is to regard it as a business entity based on the concept that maintaining each computer incurs a service fee. Such an approach would help develop the "understanding of an optimal process" that the authors refer to.

IT professionals have an obligation to educate management in how to exploit technology to benefit the company. The IT staff should carefully plan its projects and look at the big picture to find the most cost-effective solution for users.

After all, we work in the IT field not because it is easy, but because it is hard.

Hong-Lok Li
Vancouver, B.C.
lihl@ams.ubc.ca

SECURITY AT WHAT COST?

As an information security practitioner, author, and educator, I simply could not ignore the following questionable statement in Roy Want's otherwise excellent article on RFIDs ("Enabling Ubiquitous Sensing with RFID," Invisible Computing, Apr. 2004, pp. 84-86): "Despite the potential for misuse of invisible tracking, RFID's advantages far outweigh its disadvatanges."

Several questions pop up instantly: What advantages? For whom? At what cost? And whose liberties and privacy is the author ready to instantly sacrifice in search of increased streamlining, efficiency, and insecurity? Yes, insecurity, because it is largely thanks to engineers like Roy Want that we have so much cool technology with so much functionality—and so little assurance.

Why didn't the author cover the plethora of security and privacy issues associated with RFID technology? Is it because he has no answers or because that's not in the manufacturers' best interests? Instead, he makes passing mention of privacy advocates, which gives the impression that they are simply a nuisance.

When will we learn to take responsibility for opening our own Pandora's boxes? When will engineers consider more carefully the consequences of the cool technology they unleash upon us?

In the meantime, poor security practitioners and auditors are left to sort out the mess and take the blame for technology's insecurity.

Edgar Danielyan
edd@danielyan.com

SPAM AND THE LAW

Nowadays, everyone who actively uses the Internet for e-mail will inevitably have bad experiences with spam. Recent news stories describe legal action being undertaken in the US against spammers by ISPs and the FTC under the CAN-SPAM Act of 2003 (www.spamlaws.com/federal/108s877enrolled.pdf). In Europe, legislative measures against spam are also being developed under the EU Directive on Privacy and Electronic Communications. Yet, in many places such as Hong Kong, where I live, there is no antispam legislation.

It's true that the effectiveness of laws restricting spammers remains to be seen. It's also true that existing antispam laws are controversial. But this doesn't mean that we don't need antispam laws.

Many people argue that antispam laws are not going to be effective because the majority of spam doesn't originate in the countries where the laws have been enacted. That's true, but it's not a valid reason for not legislating against spamming. The Internet's global nature requires global cooperation for these legislative efforts to be fully effective. Otherwise spammers can just "escape" to another jurisdiction.

We cannot simply replace the current SMTP architecture with a tightly controlled—or even tolled—mail architecture. Instead, we must establish a norm that requires and empowers ISPs to disconnect spammers, or even spamming countries, from the Internet. A legal framework is an effective means for achieving this.

Moreover, we must not forget that spamming is not a new phenomenon restricted only to the Internet—we have junk faxes as well. Junk faxing is highly localized—with a humble computer and a modem, senders of junk faxes can create a nightmare in countries with toll-free local calls.

Does anyone really believe that antispam—or "antiunsolicited communications," to be exact—laws are not necessary?

Davy Cheung
Hong Kong
davyc@ieee.org

We welcome your letters. Send them to computer@computer.org. Letters are subject to editing for style, clarity, and length.

Travel Stories

Bob Colwell

While dozing in my customary aisle seat on a customary airplane, I heard a child's voice. Speaking with enough volume and urgency to get my sleepy attention, the child said, "Hey mister. Mister!"

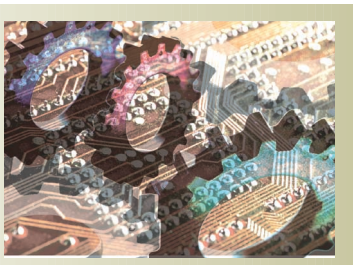
I warily opened one eye to see if I was the mister in question, hoping the answer was no. But it appeared that I was. I opened the other eye and blearily assessed the situation: The middle seat next to me was empty, and the window seat was occupied by a girl about 12 years old who was wearing an earnest, sober expression. She clearly wanted something.

"I'm bored. Let's play cards."

This was not what I wanted to hear. "I'm bored, so I'll be taking a nap, and I won't bother you during this entire flight" would have been a welcome message. Alternatively, "I'm bored, so excuse me while I let Harry Potter entertain me for the next two hours" was also okay. I tried to look polite, but disinterested. I'm not an actor, so I probably overdid it on the disinterested part, but it didn't seem to matter.

"I like poker. Do you like poker? Let's play poker. Here, I'll deal."

I tried to remember what unfortunate chain of events had led me to this predicament. Oh yeah, now I remembered. I'm an engineer. Engineering is the name they give to the act of constantly being transported from one geographic location to another. I know traveling salespeople and marketing folks will not be able to work up much sympathy, but under the circumstances, I felt entitled to a healthy dose of self-pity. Sleep or babysit someone else's kid—making that choice didn't take a lot of hand wringing.



Engineering is the name they give to the act of constantly being transported from one geographic location to another.

I said, "Cards, huh? Aren't you a little young to know how to play poker?"

She arched her eyebrow just like Spock. I half expected her to say, "That is illogical." But instead, she said, "I can beat you."

I thought that was an interesting response, so I gave her one back. "You can undoubtedly dance better than I do, and you probably sing better too. So what? That doesn't mean you and I are about to give an impromptu performance for the other passengers."

She autodeleted that answer, and said, "C'mon. Don't be a chicken. Let's make it interesting. See that guy across the aisle? I bet you the sweater he's wearing I will win the first hand."

I considered this. It reminded me of a question a friend told me he'd been asked when he interviewed for a job at the CIA. The question went like this: "You are the commander of a garrison in the middle of a desert. Suddenly, an enemy submarine surfaces and begins shooting torpedoes at you. What do you do?"

My friend initially thought they were testing his sense of humor, but quickly rejected that initial hypothesis in favor of the obvious alternative—that the question was absurd and needed an absurd answer. He said, "I would immediately send my destroyer out there and sink that sub." The interviewer smiled. My friend had passed that test. He also passed on the job.

I thought to myself, I have a potential CIA officer here. Being nice is the safest course. So I said, "Okay, you're on. If I win, I get that guy's sweater. If you win, you get his shoes." She solemnly nodded and began shuffling the deck.

I won the hand—three aces against her pair of fours. But she was undaunted. "Again. This time, I bet you the service cart and all the diet Cokes you can drink. With ice." I said, "Okay, I bet the jewelry being worn by all the people on the left side of this airplane." She won this one.

I couldn't quit now; I was too deep in the hole. Think how upset the left-side people would be to find that my bad luck or ineptitude had parted them from their jewelry. The next hand, she bet all the luggage in the overhead bins, and I bet all the laptops currently being used on the plane for business purposes (both of them). I won.

And so the fortunes of our oblivious fellow passengers waxed and waned through a few more hands. Eventually, the plane's communications systems and flight control surfaces were in play.

My card-shark friend then said,

"Since we've already bet important pieces of their aircraft, let's deal in the flight attendant and the pilot. It's only fair to give them a chance to win them back."

I pointed out that they had certain official duties to perform, such as keeping us airborne until the tires contacted the runway at the prearranged velocity. She thought for a moment, and then said, "I know—we'll deal them in and play their hands for them as best we can."

So we did. The pilot-in-absentia immediately blundered, betting the ailerons and one of the three engines on a rather poor hand. Worse, the flight attendant probably should not have bet Mt. Shasta on a pair of eights. By the time we were approaching our destination, the plane and most of the belongings on it, including some of the geographic features along the route, belonged to the 12-year-old in window seat 8F.

My opponent periodically updated the flight attendant on how she was faring along the way. The woman listened gravely each time, expressing confidence that her cards were being played fairly. Evidently, she also let the pilot know that he had been dealt into the game and was losing his shirt, because as we disembarked, he told the card player that it might take him some time to deliver the plane and asked if she would mind waiting until he could arrange it.

I shook the girl's hand and thanked her for a memorable flight. Precocious did not begin to describe this kid. I was half thinking I should have taught her some digital logic design and invited her to join my team.

THE DARKER SIDE OF TRAVELING

Then there's the time my fully guaranteed, we'll-charge-your-card-if-you-don't-show-up, expensive San Jose hotel had overbooked and refused me entrance. The overbooking part had happened to me several times before, but usually all I had to do was call my corporate travel agent, hand the phone to the hotel manager, watch him blanch as the phone emitted amazing volumes of sound, and then check into my newly available room.

But on the occasion in question, even this didn't work. They sent me elsewhere, where the scene was repeated. And those folks sent me to a third hotel, which I methodically proved over a period of two hours was not located in either Milpitas or Santa Clara. Around 3:00 a.m., I realized I was near the San Jose airport, from which my 6:30 a.m. flight would soon leave. So I drove to the rental car parking lot, climbed into the backseat of my rental car, and slept for two hours.

A welcome-to-our-country ritual can be an unwelcome experience.

A week later, I flew to San Jose again. I didn't notice anything unusual during the flight, but as I got off the plane, I observed that the pilot and a ground mechanic were engaged in a heated discussion. The pilot, his volume control set to 11, said, "If I ever find that you gave me an airplane that screwed up and dangerous to fly again, I will personally hunt you down." I quickly walked out of earshot, thinking that I did not need to know this. The pilot might not have been referring to the plane I just exited, but that possibility didn't make me feel any better.

Getting off the plane in Portland after one uneventful flight, I turned left to exit onto the Jetway, only to find that three of the passengers who had been sitting a few rows ahead of me were now spread-eagled on the floor, each with a Portland police officer's knee pressed into their back. I've occasionally seen people arrested after a flight for having sneaked into the lavatory to smoke a cigarette, but this must have been some kind of organized smoker's cabal with extra-large cigarettes. Or maybe they also smoked pipes and cigars. I sincerely hoped they had done *something* wrong because the alternative—that the Portland police were randomly selecting passengers for this special welcome—was unappealing.

The welcome I got when flying to New Zealand wasn't all that welcoming, come to think of it. We had just landed, and I was looking out the window when I noticed that we had stopped well away from the terminal. When I turned from the window, I saw that the flight attendants were donning little white surgical masks and one of them was opening the overhead bins.

The masked flight attendants walked down the aisles and, without warning, sprayed all the overhead bins and most of us with whatever was in those cans. Then they silently closed the bins and returned to their jump seats, and the plane taxied to the gate.

Later, I inquired as to the nature of the we're-so-glad-you-are-here aerosol and was told it was an insecticide so mild "it wouldn't hurt a fly." I asked the obvious question: Then why bother spraying it? The airline representative thought I was kidding and laughed heartily. I guess I'm just a funny guy.

PASSENGER PARTICIPATION

In the 1980s, the New Haven, Conn., airport was tiny and only small planes serviced it. On one trip, I booked a flight on Pilgrim Airlines from LaGuardia to New Haven and, upon arriving at the gate, discovered that things were done differently on that carrier.

We lined up to get on the plane, and a representative from the airline walked slowly down the line, looking every passenger over carefully. He apparently was rating us somehow because, for each person, he wrote something down on his list. When he got to the end of the line, all was revealed when he said, "This plane will be too heavy with all of you and your luggage. We need someone to volunteer to stay behind."

Volunteerism is a good thing. But there are times when even Mother Theresa would probably have stared at her shoes and become temporarily

incommunicado, and this was one such occasion. After a tense few minutes, when it became abundantly clear that none of us would "take one for the team," the airline rep—who turned out to be the pilot—sighed heavily and began removing bags from the checked luggage area.

This, as it turned out, was a clever stratagem, because one of the luggage owners cracked under the psychological pressure and rather crankily said, "Well, there's no sense in my going without my luggage." Eliminating that person's weight, plus his luggage, was enough to mollify the pilot. And, as our volunteer probably surmised, his sacrifice didn't generate much gratitude. As soon as the problem had been averted, everyone else promptly forgot the whole affair.

My friend Dave had a window seat for a turboprop flight on which an engine caught fire on takeoff. The pilot aborted the takeoff and informed the passengers they'd have to use the emergency exits. Dave said he got the door open and climbed out, but it was a long way to the ground. He thought he'd help the other passengers, so he stood under the emergency exit and waited.

Out the door flew briefcases, luggage, purses, and an occasional human. After everyone eventually exited the plane, some secondary concerns surfaced, such as it being January in upstate Wisconsin and that they were standing in the middle of a field.

If this had been a Bruce Willis movie, three armored cars with Rangers inside would have pulled up, fought a pitched battle with really bad people for a minute or two, and then saved all the passengers except the whiny politician and the TV newscaster. In Dave's case, 10 minutes went by before a battered old station wagon pulled up and the driver said, "Well, I can take three or four of you, and I'll come back for the rest. Looks like the fire put itself out, huh?"

A few years ago, when we were on the final approach for a landing, I glanced over at the guy sitting in the window seat, bulkhead row, and noticed that he had not "Turned Off All Electronic Devices in Preparation for Landing." In fact, he had a flight simulator running on his laptop and was landing a virtual plane. I then realized that when I looked past his laptop, out the window, the terrain changed to match what appeared on his computer screen.

Suddenly, something in one part of my brain jumped to a conclusion that the rest of my brain knew was wrong—this guy was flying the plane I was sitting in. The wrong-conclusion part of my brain screamed at my muscles to get up and guard that guy. Don't let anyone bump his arms! The rest of my brain sent concerned messages like, "What is *wrong* with you?"

Sometimes someone asks me how my flight was. I usually say, "Really, really boring. Couldn't have been better." ■

Bob Colwell *was Intel's chief IA32 architect through the Pentium II, III, and 4 microprocessors. He is now an independent consultant. Contact him at bob.colwell@comcast.net.*

JULY/AUGUST 1972

INSTRUMENTATION (p. 17). "The application of [computer system instrumentation and performance measurement] techniques … is essential to the understanding of computer system behavior. It is a necessary step towards establishing computer system design as a science. Acceptance of this notion will lead to future systems that include hardware and/or software elements which directly facilitate system instrumentation."

MINICOMPUTER (p. 58). "Honeywell Inc. has introduced a family of functional minicomputer systems known as System 700. The eight-member System 700 family is designed to expand a user's data processing system into a communications-oriented information processing network."

"The 716 central processor, with a cycle time of 775 nanoseconds for a 16-bit word, operates at more than twice the speed of the Model 316. Main memory of the 716, which is program and peripheral compatible with Series 16 central processors, ranges from 8,192 words to 32,768 words."

"Minimum systems begin at about $1,000 per month on a rental contract or can be purchased for about $30,000, depending on the system and optional equipment selected."

MICROPROCESSOR (p. 60). "Intel Corporation has introduced a second computer on a chip, an 8-bit central processor designed to handle large volumes of data. Type 8008 CPU combines with Intel RAMs, ROMs and shift registers to create MCS-8 computer systems capable of directly addressing and retrieving as many as 16,000 8-bit bytes stored in the memory devices.

"The CPU is a P-channel silicon-gate MOS circuit containing an 8-bit parallel adder, six 8-bit data registers, an 8-bit accumulator, two 8-bit temporary registers, four flag bits and eight 14-bit address registers

PREP SCHOOL (p. 62). "Using an IBM System/3 Model 6, the Creighton Preparatory School in Omaha, Nebraska, founded in 1878 and conducted by the Jesuits, is training young men in college oriented educational programs. The new ideas include some innovative administrative shortcuts as well as classroom instruction in computer usage at the high school level."

"Father Robert Worman, who teaches the two computer classes to 27 students, stated that 'The keyboard input and the RPG Language of the System/3 are ideal for teaching at the high school level. Young men at this age have a facile memory and pick up the computer language just like another spoken language.'"

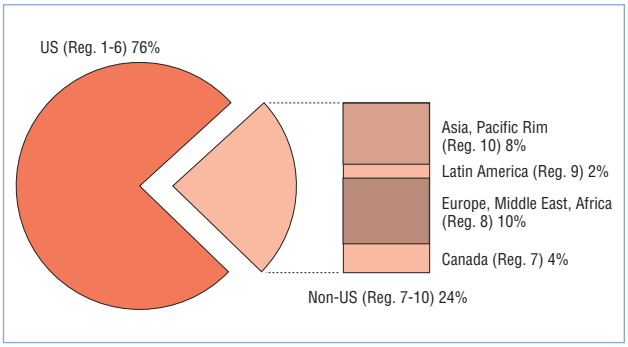
TOXICOLOGY (p. 63). "Representatives of the Toxicology Information Program of the National Library of Medicine, National Institute of Health, U.S. Department of Health, Education and Welfare, and Informatics, Inc. have performed on-line demonstrations of TOXICON, a service sponsored by the Toxicology Information Program and operated by Informatics.

"Utilizing a Cathode Ray Tube (CRT) terminal, commands were issued via a keyboard to a computer in Washington D.C. to search files of citations and abstracts of published toxicology and pharmacology studies of drugs, pesticides, and environmental pollutants and to provide information in the specific areas. The response from the computer took a matter of seconds.

"According to Dr. Walter F. Bauer, president of Informatics, TOXICON is an important example of how on-line communications systems available to the public and tied to computerized data collections of governmental agencies can operate to the public's benefit."

JULY 1988

THE COMPUTER SOCIETY (p. 4). "While our transnational tradition has long been recognized, new trends are emerging that make it more evident. Principal among these is that the annual rate of growth of non-US society members has begun to exceed the US membership growth rate. As shown in the accompanying chart, approximately one quarter of all the society's members (all grades) are residents of IEEE Regions 7-10. The biggest international segments are Region 8 (Europe, Middle East, and Africa) and Region 10 (Asia and the Pacific Rim)."



HARDWARE VERIFICATION (p. 18). "Industry, abandoning its initial skepticism, is becoming more and more interested in formal verification, since it can guarantee correct designs and shave costly development time. Some major European manufacturers plan to include in their private CAD systems some of the formal verification tools currently under development.

"If formal verification keeps in touch with the latest developments in VLSI, so computer-aided design does not lag behind design, both fields will benefit and contribute significantly to the advancement of computer science."

OPEN SOFTWARE (p. 62). "Seven leading computer companies have established an international foundation to provide a completely open software environment designed to

32 & 16 Years Ago

facilitate customer use of computers and software from many vendors.

"The Open Software Foundation will develop a software environment, including application interfaces, advanced system extensions, and a new operating system using the specifications for X/Open and Portable Operating System Interface for Computer Environments (Posix) as the starting point. The Posix standard, developed by the Computer Society's Technical Committee on Operating Systems and closely related to the Unix system, specifies how software should be written to run on computers from different vendors."

SEMICONDUCTOR MANUFACTURING (p. 71). "Sematech has designated five university Centers of Excellence to support the semiconductor manufacturing research consortium's objective of restoring US leadership in semiconductor manufacturing technology."

"The centers were selected by a team of 36 leading technical experts in the semiconductor industry. The experts based their review of proposals and selection of centers on the quality of the research program offered, its relevance to Sematech's needs, and the nature of the program resources that would be made available at each center."

RISC MICROPROCESSORS (p. 86). "Motorola has announced the 88000 family, a new product line of reduced-instruction-set-computer microprocessors. The family includes the 88100 RISC Microprocessor and 88200 Cache/Memory Management Unit.

"According to the company, the 88000 series architecture couples pipelined floating-point and integer units on a single chip. It also incorporates a technique called scoreboarding, which reportedly allows the processor to perform as many as 11 operations concurrently.

"The 88100 integrates an integer and two floating-point units. It supports six special function units. The 88100 contains 165,000 transistors.

32-BIT ARCHITECTURE (p. 87). "Intel has announced a 32-bit microcomputer architecture that integrates reduced-instruction-set-computer design techniques. According to the company, the core 80960 32-bit architecture has parallelism and modular features for increased performance levels and development of market-specific, embedded control processors."

"On-chip functions of the 80960KB include 32 × 32-bit registers, a floating-point unit, a 512-byte instruction cache, a stack frame cache, and a 32-bit multiplexed burst bus. The chip also has an interrupt controller with 256 interrupt vectors and 32 levels of interrupt priority."

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

Open Source Databases Move into the Marketplace

Linda Dailey Paulson

A growing number of small and midsized organizations are using databases for relatively simple functions not requiring the complex, expensive software that vendors typically sell to large companies and government agencies.

The smaller organizations are thus increasingly turning to open source databases, which tend to cost less than their proprietary counterparts, such as those sold by IBM, Microsoft, and Oracle.

According to analyst Paul Kirby of AMR Research, a market analysis firm, a survey of and interviews with 140 IT managers indicated that open source databases will gain widespread acceptance by 2006.

The major open source databases include Firebird, MaxDB, MySQL, PostgreSQL, and Sleepycat Software's Berkeley DB.

Companies such as Google and Travelocity use these products as databases in their applications. Nokia uses Berkeley DB for short-message-service storage and retrieval. Westone Laboratories, a hearing aid manufacturer, uses MySQL as its patient-information database.

"There are quite a few serious enterprise-level adoptions, but not on the scale of proprietary databases, of course" said analyst Joe McKendrick with Evans Data, a market research firm.

Although growing in popularity, open source databases face several challenges to marketplace success. For example, they frequently lack some complex capabilities that large organizations may want. And the databases also have trouble scaling in terms of data volume and number of users.

ABOUT OPEN SOURCE DATABASES

After years as the subject of research, open source databases are beginning to gain commercial acceptance. As this occurs, different models of development, distribution, licensing, and availability have emerged, as Figure 1 shows.

Development

In general, open source software is developed, tested, debugged, and/or improved through public collaboration, and its source code is openly available for use or modification as users see fit.

MySQL and Sleepycat, for example, have programming staffs for developing the software but release the code for the open source community to work on debugging and testing.

PostgresSQL and MaxDB are freely available online. And Firebird is a free database that its developers call "a commercially independent project of C and C++ programmers, technical advisors, and supporters."

Organizational models

The open source databases also have different organizational models. For example, MySQL AB and Sleepycat are private companies. They sell licenses, support, and consulting to commercial users of their databases and make their software free for open source projects.

MaxDB, MySQL's version of software vendor SAP's SAP DB, is also freely available for open source use. MySQL makes money on commercial uses of MaxDB via fees for licenses, training, support, and consulting.

The PostgreSQL Global Development Group (http://developer.postgresql.org) is an organization of companies and individuals driving development of PostgreSQL.

The nonprofit FirebirdSQL Foundation (http://www.firebirdsql.org/ff/foundation), a volunteer effort sponsored by various companies including BroadView Software, oversees Firebird. The Firebird Project—an independent group of developers and other supporters—handles the database's distribution and technical development.

The technologies

Technically, the open source databases take different approaches.

MaxDB and MySQL are relational databases that support commonly used languages such as C++ and Java. They can also handle binary large objects, which are typically big image or sound files. The storage of large multimedia objects, such as films, is an important issue for databases.

Zack Urlocker, MySQL's vice president of marketing, said that both databases are suited for a wide range of applications but that MySQL is more flexible while MaxDB is best for applications requiring SAP-certified databases.

	Open source Sleepycat Berkeley DB	Closed source database	Open source Linux
Development team	All developers work for Sleepycat	All developers work for company	Thousands of volunteer developers
Distribution	Distributed without charge for all open source projects; license fees for closed source projects	Distribution usually requires reseller agreement	Freely distributed
License	Separate licenses for use in open source and closed source projects	Commercial license	Open source license
Availability	Free download	Must purchase first or test limited product	Free download

Source: Sleepycat

Figure 1. As open source databases have become more popular, different models of development, distribution, licensing, and availability have emerged. This chart compares the Berkeley DB open source database in these categories with closed source databases and the open source Linux operating system.

Firebird is also a relational database management system, explained Helen Borrie, the Firebird Project's administrator. Based on Borland Software's InterBase 6 database, Firebird is programmed in C++ and uses the Structured Query Language (SQL) for getting information from and updating databases.

Borrie said the system scales well and is good for use with multiuser applications in banking and government, as well as with small-business LANs. Also, she said, "Its tolerance to high-volume read-write activity makes it attractive as the data layer for interactive Web applications."

Sleepycat CEO Mike Olson said Berkeley DB is an embedded database that uses APIs instead of SQL. Rather than restricting users to SQL, Berkeley DB lets users work via APIs that operate directly on the database and its records.

The database can be embedded in servers, networking hardware, and handheld devices. A programmer can use any number of languages to write an application, which then uses simple commands to store or retrieve data.

For example, Berkeley DB supports Perl, a cross-platform programming language, and Python, an interpreted, interactive, object-oriented programming language. Both Perl and Python are open source.

The database uses the PostQuel query language. PostgreSQL has a GUI and interfaces for C, Embedded C, Java, Python, and the Tcl interpreted script language.

More capabilities and uses

AMR's Kirby said that because open source databases are new, developers started by implementing simpler features. And some companies have used the databases only for simpler tasks, he said, such as fetching data for Web sites, because they haven't trusted these relatively new products with more complex and important tasks.

However, proponents say open source databases now offer strong performance and many high-end features and can compete with their proprietary counterparts on capabilities as well as price.

Sleepycat's Olson said, "Berkeley DB is a high-performance database with advanced features that can handle tables up to 256 Tbytes and up to 2 Gbytes of data in a single record. The assumption that open source databases cannot compete with closed source databases on features is incorrect. We support replication, hot standby, load balancing, and full transaction semantics."

MySQL's Urlocker said, "I think there are a lot of myths out there." He explained that MySQL is used for such heavy-duty work as warehousing between four and five Tbytes of data, crunching large data volumes for genomic research, and running servers for busy Web sites.

However, Kirby said, some of the open source databases that offer advanced features have not implemented them as robustly as the makers of high-end proprietary databases.

"I think it's fair to say we do not have all the bells and whistles of the high-end Oracle or [IBM] DB2 servers. So if you want grid computing or an XML database, those are features we don't have," Urlocker said. "But for 80 percent of the applications, these features are not needed."

Meanwhile, noted Tim O'Reilly, founder and president of O'Reilly & Associates, open source databases are being used in a growing number of settings.

Costs and licensing advantages

Open source databases are typically less expensive to sell because they generally have little or no marketing or research-and-development costs.

And for users, said Firebird's Borrie, "Open source database-management systems are less expensive to adopt if there are no software or license fees to pay."

Traditional database companies charge $40,000 or more per server processor for their software. The most expensive open source database, MaxDB, costs $1,500 per processor.

This is a critical factor because, AMR's Kirby noted, among surveyed companies anticipating they will evaluate new database technology within the next two years, 42 percent were motivated primarily by cost.

Some proprietary-database vendors

have been lowering their prices recently. Jeff Jones, director of strategy for IBM Software Group's Information Management Solutions unit, said his company is dropping prices to attract users who want less expensive databases, not in response to challenges from open source vendors.

Nonetheless, he added, the open source movement in general is getting companies to think more about cost effectiveness.

Even if vendors reduce their prices, Kirby said, they probably won't drop them enough to compete with open source databases on cost alone.

Open source products also offer favorable licensing terms. There are about 20 types of open source licenses. They let individuals and organizations freely work with the licensed applications in open-source products without the need to secure usage permissions, which can be a costly and lengthy process.

Vendors that want to include an open source database in a proprietary software product can secure a license that lets them do so, often at lower cost than a proprietary database's license.

MySQL and Sleepycat each offer two licenses, one when the databases are used in open source software and one when they are used in proprietary software.

UPGRADING THE DATABASES

To better meet user demands, open source database vendors are upgrading their products' capabilities.

One important new feature is *failover*, a backup mode in which a secondary system assumes a system component's functions when the latter becomes unavailable through failure or scheduled downtime. This increases data availability, which is important for mission-critical applications.

MySQL Cluster—based on cellular-phone vendor Ericsson's NDB Cluster software—runs up to 64 nodes in parallel and automatically replicates data across them. The system monitors each node at all times, Urlocker said. If the one in use fails, he explained, the system redirects queries in milliseconds to other nodes.

According to Olson, Sleepycat has coupled failover with fault tolerance in Berkeley DB.

MySQL and PostgreSQL have added support for *stored procedures*, another feature frequently found in high-end proprietary databases. A stored procedure is a precompiled operation or query stored in the database server and accessible for use by clients.

Open source database vendors are upgrading their products' capabilities.

Meanwhile, vendors are adding graphical management tools to open source databases, which make the applications easier to work with, particularly for users unfamiliar with text commands. For example, MySQL has added an administration-console GUI.

Another important feature, Olson noted, is the ability to work with XML. Organizations frequently use databases to generate Web pages, many of which are written in XML. Sleepycat recently added native XML storage and query-processing capabilities.

CAUTIONARY CONSIDERATIONS

Commercial success has escaped open source databases in the past. For example, Great Bridge closed its doors in 2001 after failing to successfully market a PostgreSQL-based database. And Linux vendor Red Hat, which began work on an open source database in 2001, terminated its project the following year.

While some industry sources may look at these events as bad signs, O'Reilly contended they are simply a normal part of market evolution, in which some early versions of new products don't succeed.

Nonetheless, according to the recent AMR study, open source databases will have to better scale up to hold more data and work with a greater number of concurrent users if they are to win market share.

And although developers are upgrading the databases' capabilities, some still lack highly complex functions that are important for large-scale corporate use, such as analytics, which enable sophisticated data analysis.

Open source databases haven't made a significant dent in the database marketplace yet. The Aberdeen Group, a market research firm, says open source products accounted for about $100 million of the $10.5 billion in revenue generated by the database market in 2003.

However, a growing number of companies are integrating open source databases into their products, which may increase the databases' market share. For example, Sun Microsystems uses Berkeley DB in its directory server.

A recent survey by Evans Data found that the number of applications that run MySQL grew 30 percent between December 2002 and December 2003, compared to 6 percent growth for Microsoft's SQL Server. Evans' McKendrick said four of 10 companies surveyed use an open source database, indicating "the end user companies are getting comfortable with open source technology."

Growth has been helped by economic pressures on companies to save money and, Urlocker noted, IT spending constraints.

According to McKendrick, open source databases will continue to thrive, especially as data-storage needs mushroom. ■

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

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

Not A Member Yet?

Here Is What You're Missing...

Distance Learning

Members of the IEEE Computer Society enjoy FREE access to a comprehensive distance learning program for computer professionals. Join today and take any or all of 100 online course titles through our Distance Learning Campus. Subjects include Java, project management, Cisco networks, UNIX, Windows, XML, Oracle, SQL, and more.

Magazines and Journals

A FREE subscription to *Computer* magazine is included with your membership. You may also subscribe to other publications in your area of expertise at member discounts. Or subscribe to the *IEEE Computer Society Digital Library*—21 periodicals and 1200+ conference proceedings—for a complete online resource.

Conferences and Workshops

Enhance your knowledge and share practical experiences at more than 150 conferences, workshops, and symposia held each year worldwide. Members save at least 25% on registration fees and get advance notice of the meetings.

Other Valuable Benefits

- FREE membership in your local chapter
- FREE e-mail alias of Your.Name@computer.org
- FREE membership in up to four of 40+ Technical Committees
- FREE membership in 160+ Standards Working Groups
- Member discounts on hundreds of books and conference proceedings

Join the IEEE Computer Society Today!

Complete the adjacent membership application today. For fastest service, apply online now at

http://computer.org/join

IEEE

IEEE COMPUTER SOCIETY

2004 IEEE Computer Society Professional Membership/Subscription Application

Membership and periodical subscriptions are annualized to and expire on 31 December 2004.
Pay full or half-year rate depending upon the date of receipt by the IEEE Computer Society as indicated below.

Membership Options*

All prices are quoted in U.S. dollars

	FULL YEAR Applications received 16 Aug 03 - 29 Feb 04	HALF YEAR Applications received 1 Mar 04 - 15 Aug 04
1 I do not belong to the IEEE, and I want to join **just** the Computer Society	$ 99 ❑	$50 ❑
2 I want to join **both** the Computer Society and the IEEE:		
I reside in the United States	$189 ❑	$95 ❑
I reside in Canada	$170 ❑	$85 ❑
I reside in Africa/Europe/Middle East	$166 ❑	$83 ❑
I reside in Latin America	$159 ❑	$80 ❑
I reside in Asia/Pacific	$160 ❑	$80 ❑
3 I already belong to the IEEE, and I want to join the Computer Society. (IEEE members need only furnish name, address, and IEEE number with payment.)	$ 42 ❑	$21 ❑

Are you now or were you ever a member of the IEEE?
Yes ❑ No ❑ If yes, provide member number if known: _____

Add Periodicals**

	ISSUES PER YEAR	FULL YEAR Applications received 16 Aug 03 - 29 Feb 04 PRINT	ELECTRONIC	COMBO	HALF YEAR Applications received 1 Mar 04 - 15 Aug 04 PRINT	ELECTRONIC	COMBO
IEEE Computer Society Digital Library **BEST DEAL**	n/a	n/a	$109 ❑	n/a	n/a	$55 ❑	n/a
Computing in Science and Engineering	6	$42 ❑	$34 ❑	$55 ❑	$21 ❑	$17 ❑	$28 ❑
IEEE Computer Graphics and Applications	6	$39 ❑	$31 ❑	$51 ❑	$20 ❑	$16 ❑	$26 ❑
IEEE Design & Test of Computers	6	$37 ❑	$30 ❑	$48 ❑	$19 ❑	$15 ❑	$24 ❑
IEEE Intelligent Systems	6	$37 ❑	$30 ❑	$48 ❑	$19 ❑	$15 ❑	$24 ❑
IEEE Internet Computing	6	$39 ❑	$31 ❑	$51 ❑	$20 ❑	$16 ❑	$26 ❑
IT Professional	6	$40 ❑	$32 ❑	$52 ❑	$20 ❑	$16 ❑	$26 ❑
IEEE Micro	6	$37 ❑	$30 ❑	$48 ❑	$19 ❑	$15 ❑	$24 ❑
IEEE MultiMedia	4	$35 ❑	$28 ❑	$46 ❑	$18 ❑	$14 ❑	$23 ❑
IEEE Pervasive Computing	4	$41 ❑	$33 ❑	$53 ❑	$21 ❑	$17 ❑	$27 ❑
IEEE Security & Privacy	6	$41 ❑	$33 ❑	$53 ❑	$21 ❑	$17 ❑	$27 ❑
IEEE Software	6	$44 ❑	$35 ❑	$57 ❑	$22 ❑	$18 ❑	$29 ❑
IEEE/ACM Transactions on Computational Biology and Bioinformatics **NEW**	4	$35 ❑	$28 ❑	$46 ❑	$18 ❑	$14 ❑	$23 ❑
IEEE/ACM Transactions on Networking†	6	$44 ❑	$33 ❑	$55 ❑	$22 ❑	$17 ❑	$28 ❑
IEEE Transactions on:							
Computers	12	$41 ❑	$33 ❑	$53 ❑	$21 ❑	$17 ❑	$27 ❑
Dependable and Secure Computing **NEW**	4	$31 ❑	$25 ❑	$40 ❑	$16 ❑	$13 ❑	$20 ❑
Information Technology in Biomedicine†	4	$35 ❑	$25 ❑	$44 ❑	$18 ❑	n/a	$22 ❑
Knowledge and Data Engineering	12	$43 ❑	$34 ❑	$56 ❑	$22 ❑	$17 ❑	$28 ❑
Mobile Computing	4	$30 ❑	$24 ❑	$39 ❑	$15 ❑	$12 ❑	$20 ❑
Multimedia†	6	n/a	n/a	$38 ❑	n/a	n/a	n/a
NanoBioscience†	4	$30 ❑	$24 ❑	$38 ❑	$15 ❑	n/a	$19 ❑
Parallel and Distributed Systems	12	$40 ❑	$32 ❑	$52 ❑	$20 ❑	$16 ❑	$26 ❑
Pattern Analysis and Machine Intelligence	12	$44 ❑	$35 ❑	$57 ❑	$22 ❑	$18 ❑	$29 ❑
Software Engineering	12	$38 ❑	$30 ❑	$49 ❑	$19 ❑	$15 ❑	$25 ❑
Visualization and Computer Graphics	6	$34 ❑	$27 ❑	$44 ❑	$17 ❑	$14 ❑	$22 ❑
VLSI Systems†	12	n/a	n/a	$28 ❑	n/a	n/a	$14 ❑
IEEE Annals of the History of Computing	4	$31 ❑	$25 ❑	$40 ❑	$16 ❑	$13 ❑	$20 ❑

Choose PRINT for paper issues delivered via normal postal channels.
Choose ELECTRONIC for 2004 online access to all issues published from 1988 forward.
Choose COMBO for both print and electronic.

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

_____
Expiration date (month/year)

_____
Signature

USA-only include 5-digit billing zip code
☐☐☐☐☐

* Member dues include $17 for a 12-month subscription to *Computer*.

** Periodicals purchased at member prices are for the member's personal use only.

*** Canadian residents add 15% HST or 7% GST to total. AL, AZ, CO, DC, NM, and WV add sales tax to all periodicals. GA, IN, KY, MD, and MO add sales tax to print and combo periodicals. NY add sales tax to electronic periodicals. European Union residents add VAT tax to electronic periodicals.

**** 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.

† Not part of the IEEE Computer Society Digital Library. Electronic access is through www.ieee.org/ieeexplore.

For fastest service, apply online at http://computer.org/join

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
PO Box 3014
Los Alamitos, CA 90720-1314 USA
Phone: +1 714 821 8380
Fax: +1 714 821 4641
E-mail: help@computer.org

Allow up to 8 weeks to complete application processing. Allow a minimum of 6 to 10 weeks for delivery of print periodicals.

IF4G

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

APPLICATION MUST BE SIGNED

NOTE: In order for us to process your application, you must complete and return both sides of this form.

IEEE

IEEE COMPUTER SOCIETY

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, 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

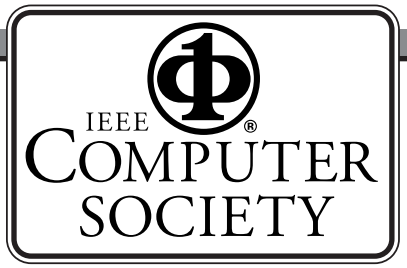
1. General and corporate management
2. Engineering management
3. Project engineering management
4. Research and development management
5. Design engineering management - analog
6. Design engineering management - digital
7. Research and development engineering
8. Design/development engineering - analog
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 _____



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
The IEEE Computer Society's Web site, at **www.computer.org**, offers information and samples from the society's publications and conferences, as well as a broad range of information about technical committees, standards, student activities, and more.

OMBUDSMAN Members experiencing problems—magazine delivery, membership status, or unresolved complaints—may write to the ombudsman at the Publications Office or send an e-mail 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 the Publications Office:

- 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)

To check membership status or report a change of address, call the IEEE toll-free number, +1 800 678 4333. Direct all other Computer Society-related questions to the Publications Office.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, industry trends, and product reviews.

Periodicals. The society publishes 12 magazines and 10 research transactions. Refer to membership application or request information as noted at left.

Conference Proceedings, Tutorial Texts, Standards Documents. The Computer Society Press publishes more than 160 titles every year.

Standards Working Groups. More than 200 groups produce IEEE standards used throughout the industrial world.

Technical Committees. Thirty TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Is MIMO the Future of Wireless Communications?

George Lawton

Wireless-system designers are faced with numerous challenges, including limited availability of radio-frequency spectrum and transmission problems caused by such factors as fading and multipath distortion.

Meanwhile, there is increasing demand for higher data rates, better-quality service, fewer dropped calls, and higher network capacity. Meeting these needs requires new techniques that improve spectral efficiency and network links' operational reliability.

Multiple-input-multiple-output technology promises a cost-effective way to provide these capabilities. MIMO uses antenna arrays at both the transmitter and receiver. Algorithms in a radio chipset send information out over the antennas. The radio signals reflect off objects, creating multiple paths that in conventional radios cause interference and fading. But MIMO sends data over these multiple paths, thereby increasing the amount of information the system carries. The data is received by multiple antennas and recombined properly by other MIMO algorithms.

This technology promises to let engineers scale up wireless bandwidth or increase transmission ranges.

MIMO is an underlying technique for carrying data. It operates at the physical layer, below the protocols

used to carry the data, so its channels can work with virtually any wireless transmission protocol. For example, MIMO can be used with the popular IEEE 802.11 (Wi-Fi) technology.

For these reasons, MIMO eventually will become the standard for carrying almost all wireless traffic, according to Greg Raleigh, president and CEO of wireless vendor Airgo Networks.

MIMO still must prove itself in large-scale, real-world implementations, and it must overcome several obstacles to its success, including energy consumption, cost, and competition from similar technologies.

Nonetheless, said Craig Mathias, an analyst with The Farpoint Group, a wireless communications and computing consultancy, "We think it will become a core technology in wireless systems. It is really the only economical way to increase bandwidth and range."

MIMO

Numerous companies—including Airgo, Intel, and Lucent Technologies—have announced plans to release MIMO-based products. This summer, Airgo plans to release the first MIMO chips for incorporation in wireless LAN cards.

MIMO background

MIMO was originally conceived in the early 1970s by Bell Labs engineers trying to address the bandwidth limitations that signal interference caused in large, high-capacity cables. At the time, however, the processing power necessary to handle MIMO signals was too expensive to be practical.

Advances to and cost reductions in signal-processing technology, coupled with increased demands to overcome the limits of existing mobile communications approaches, have since led researchers to reconsider MIMO for wireless systems.

How it works

Signals in a wireless system frequently reflect off objects en route to the recipient and bounce along different paths. At various points, the signals become out of synch, thereby scrambling the received transmission and decreasing bandwidth, creating a problem called *multipath distortion*.

As Figure 1 shows, MIMO takes advantage of this situation by sending a single transmission from two or more antennas to bounce along multiple paths to a receiver. Putting data on multiple signal paths increases the amount of information a system can carry and the number of users it can serve. In addition, this approach lets a system divide a single data set into parts that are sent over multiple paths in parallel. This lets the system handle the information faster than approaches that send data over a single path.

For example, first-generation MIMO products would double IEEE 802.11's theoretical maximum data rate from 54 to 108 Mbits per second.

The nature of the signals on each path is changed slightly based on the different antennas from which they are sent, the spacing of the antennas, and the type of interference the signals en-

counter. The recipient's system analyzes this information via matrix-manipulation signal-processing technology—which cross-correlates the signals—to detect their various paths and reconstitute them properly. This process also reduces the effects of interference.

Moreover, by spreading a transmission signal across multiple paths, MIMO increases the chance that any given path will reach the destination, which improves link reliability.

In addition, MIMO systems can choose from the multiple antennas they work with to use those with the clearest signals. This reduces error rates and improves communication quality.

MIMO can also increase the effective transmission range of the overriding wireless technology being used. By using clearer signals and minimizing the effects of interference, MIMO signals can be resolved over longer distances than technologies whose effective ranges are reduced by noise and signal diffusion.

It's theoretically possible to continue increasing data rates and transmission ranges by adding antennas to a system. In practice, though, engineers are limited by the nature of the multipath environment, such as the number and nature of obstacles encountered, and the increased processing power required to handle the extra work generated by additional antennas.

Gee Rittenhouse, vice president of wireless research for Lucent Technologies' Bell Labs, said installing up to four antennas per transmitter or receiver in suburban environments and 16 in urban environments, where a denser user population would require denser antenna concentrations, is practical.

MIMO'S ADVANTAGES

MIMO's higher speeds are critical for letting wireless networks handle data-intensive multimedia files. The increased bandwidth also lets wireless networks serve more users at a given data rate than they could without MIMO.

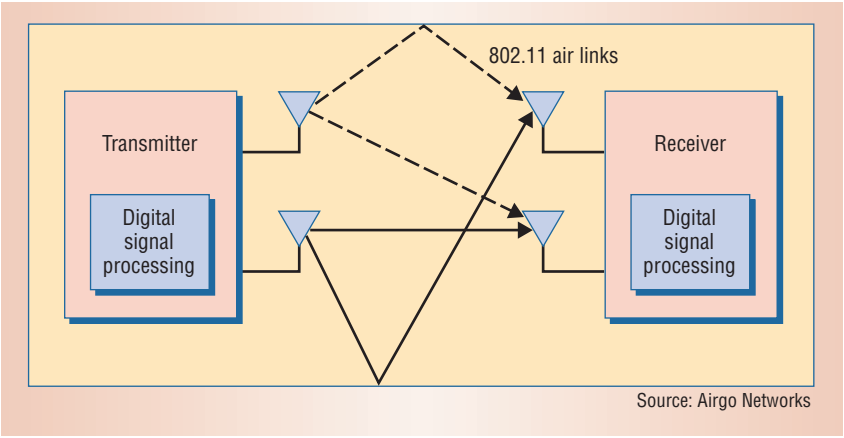


Figure 1. MIMO sends a transmission from multiple antennas to bounce over multiple paths to a receiver, in this case using the overriding IEEE 802.11 wireless LAN technology. MIMO has several benefits. For example, it increases the amount of information a system can carry, which allows it to serve more users. In addition, a system can divide a single data set into parts that it sends over multiple paths in parallel, thereby handling the information more quickly than data sent over a single path.

And the increased range of MIMO LANs' base stations would let large businesses serve their entire organization with fewer stations, thereby saving them money.

Because the technology reduces the effects of interference and can focus on better-quality signals, MIMO networks use less radio-transmission power than other wireless networks, so there is less battery drain on portable systems and less chance of interference with or from other systems.

In addition, because MIMO sends transmissions along multiple paths, most of the signals can avoid objects and other sources of interference that cause fading and interruptions.

And senders can adjust the power and phase given to antennas to steer signals toward the paths with the best transmission quality. More precise steering could minimize the interference a sender causes or receives, explained Andrea Goldsmith, associate professor at Stanford University.

According to Goldsmith, MIMO's signaling properties could also help create more robust wireless security. It would be difficult for hackers to set up their receivers to properly receive all of the signals that have been broken up and sent via multiple antennas along different paths.

IMPLEMENTING MIMO

MIMO has several important implementation issues. For example, users can achieve modest performance gains by implementing MIMO only at the transmitter, but enabling both transmitters and receivers to take advantage of the technology yields much greater improvements.

MIMO chips

MIMO is implemented via chips. In addition to typical wireless-processing cores, the chips contain ports for multiple antennas and matrix-manipulation processing elements optimized to process MIMO signals.

Developers can implement MIMO in base stations or LAN cards. For a big organization, integrating MIMO into a base station is much less expensive than upgrading hundreds of LAN cards.

Vendors such as Airgo are already incorporating MIMO chips into Wi-Fi LAN cards. Airgo's Raleigh said that the early market will be for WLANs but that companies eventually will integrate MIMO into almost all types of radio-based wireless equipment.

New implementation approaches

Researchers are now focusing on two popular coding schemes for using MIMO to carry traffic: orthogonal frequency-division multiplexing, supported by companies such as Airgo and Lucent, and code-division multiple access.

OFDM increases bandwidth and data capacity by splitting broad channels into multiple narrowband channels—each using a different frequency—that can then carry different parts of a message simultaneously. To maximize capacity, the channels are spaced closely together but avoid interference because neighboring channels are orthogonal to one another and thus have no overlap.

CDMA is a type of multiplexing that lets multiple signals occupy a single transmission channel, optimizing the use of available bandwidth. The system varies a transmitted signal's frequency according to a defined pattern, known as a *code*, so that only a receiver whose frequency response is programmed with the same code can successfully intercept it. Thus, signals intended for multiple recipients can be coded differently and carried at the same time on the same channel.

HURDLES TO CLEAR

Despite its promise, MIMO still faces several challenges.

Technical challenges

Designing MIMO systems, which send signals over multiple transmission paths, is a challenge, particularly because most wireless engineers have worked only on systems designed to use one transmission path, according to Raleigh.

Also, MIMO has worked well in a laboratory environment between two fixed nodes. However, said Stanford's Goldsmith, there are questions about how well it will work in a real-world environment between mobile nodes.

She added, "Many of the algorithms and the performance gains assume that you know the nature of the transmission channel perfectly or almost perfectly." In real-world mobile environments, she said, the nature of the channel changes regularly as users move about.

MIMO's biggest technical challenge may be the increased processor-energy consumption caused by the processing complexity required to handle signals traveling multiple paths between antenna arrays.

Despite its promise, MIMO still faces challenges.

First-generation MIMO vendors are reducing this problem by using small antenna arrays. In addition, researchers are working on CPU power efficiency.

Marketplace challenges

There are concerns that businesses and consumers won't pay the added cost of incorporating MIMO into their networks.

Raleigh estimated that the first MIMO chips will add about $20 to a LAN card's price but that more efficient integration and higher sales volumes will reduce this cost over time.

In some applications in which MIMO reduces the number of necessary base stations, the savings justifies the cost. But in other cases, such as consumer applications, MIMO will make sense only when its cost drops significantly. Consumer applications would use MIMO in WLANs to connect laptops and multimedia devices to the Internet and to one other.

For now, though, Mathias said, MIMO doesn't make sense for most home and small-office deployments because their Internet connections can't take advantage of the technology's increased bandwidth. However, he added, MIMO could benefit new deployments at larger organizations with higher-bandwidth networks.

Other multi-antenna technologies may also threaten MIMO. For example, ArrayComm has developed a proprietary system that uses multiple antennas to improve the range of cellular-system base stations without requiring changes to existing mobile phones.

This technology is already being deployed on mobile systems in Japan and China, with other implementations on the way, said Adam Kerr, ArrayComm's vice president of engineering research.

Already, MIMO has become part of the IEEE 802.16d wireless networking standard. Numerous vendors, such as Airgo and Lucent, are promoting MIMO as the IEEE's next 802.11 standard, 802.11n, which the organization expects to complete by 2006. In addition, the Third Generation Partnership Project, a collaboration of telecommunications standards organizations, is evaluating MIMO for cellular networks.

Some sources say MIMO holds promise for cellular networks but is limited by the heavy cost of upgrading base stations. Farpoint's Mathias said, "MIMO in cell phones is a little more difficult proposition because there are so many cell sites without it."

According to Stanford's Goldsmith, MIMO may not be widely used in cellular phones because the cost of implementing it outweighs the benefits.

However, Mathias predicted wide deployment of MIMO in WLANs within three years. "I think part of it is simple marketing," he explained, "letting people know that this is not exotic technology and that it works." ■

George Lawton *is a freelance technology writer based in Brisbane, California. Contact him at glawton@glawton.com.*

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

GET CERTIFIED

IEEE Computer Society Certified Software Development Professional

CSDP

CERTIFIED SOFTWARE DEVELOPMENT PROFESSIONAL PROGRAM

Apply now for the 1 September—30 October test window.
(Deadline to apply: 15 August)

Doing Software Right

- Demonstrate your level of ability in relation to your peers
- Measure your professional knowledge and competence

Certification through the CSDP Program differentiates between you and other software developers. Although the field offers many kinds of credentials, the CSDP is the only one developed in close collaboration with software engineering professionals.

"The exam is valuable to me for two reasons:
One, it validates my knowledge in various areas of expertise within the software field, without regard to specific knowledge of tools or commercial products...
Two, my participation, along with others, in the exam and in continuing education sends a message that software development is a professional pursuit requiring advanced education and/or experience, and all the other requirements the IEEE Computer Society has established. I also believe in living by the Software Engineering code of ethics endorsed by the Computer Society. All of this will help to improve the overall quality of the products and services we provide to our customers..."

— Karen Thurston,
Base Two Solutions

Visit the CSDP web site at www.computer.org/certification
or contact certification@computer.org

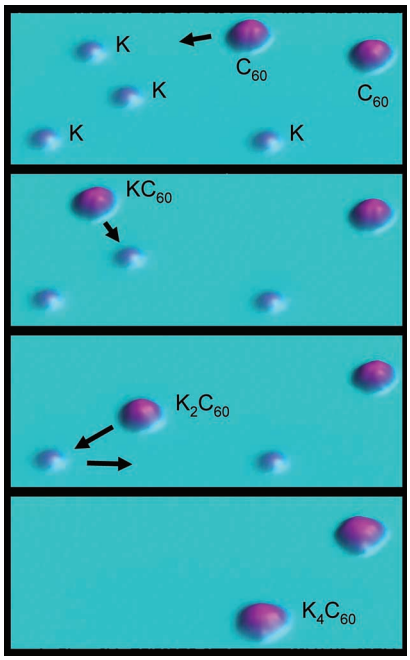
IEEE

IEEE COMPUTER SOCIETY

The New Dope on Semiconductor Doping

A University of California, Berkeley, researcher has developed a way to apply semiconductor dopants at the atomic level. Controlling the amount of dopant applied is increasingly important as the size of wires, diodes, transistors, switches, and other semiconductor elements approaches molecular scale.

During semiconductor fabrication, manufacturers typically add dopants to semiconductor materials, such as silicon, to change their performance by altering their electrical properties. P-type doping adds elements such as boron or indium to remove electrons. N-type doping inserts elements such as arsenic, phosphorous, or potassium to add electrons.



These photos show how a UC Berkeley researcher's new technique enables carbon molecules to absorb potassium atoms as dopants. In the past, chip makers frequently added dopants, which improve a material's performance by changing its electrical properties, in bulk because their exact quantity or placement wasn't important. As semiconductor elements get smaller, being able to add tiny amounts of dopants in a controlled manner becomes critical.

Currently, chip makers frequently add dopants in bulk to semiconductor materials. However, as the elements become smaller, the amount and placement of doping must become more precise. "If you make your electrical components small enough, the presence or absence [or placement] of a single dopant atom influences the properties of the devices," said Michael F. Crommie, professor of physics at UC Berkeley.

In response, Crommie developed a technique for attaching dopant atoms, one at a time, to individual molecules, permitting the tight control and fine tuning of their electronic properties. Conventional tools have not offered such precise control.

Crommie and his group used the probe of a scanning tunneling microscope—whose sharp metal needle reads a surface when voltage is applied—to move large buckminsterfullerene carbon molecules across very cold, highly polished silver crystal toward potassium dopant atoms. When a molecule gets close enough, it sucks up an atom.

Crommie conducted his experiment in a vacuum chamber at 7 degrees Kelvin (-266 degrees Celsius), which is close to absolute zero. The extreme cold is critical because it keeps the atoms from moving around, which makes it easier for the molecules to absorb them.

Using extreme cold would not necessarily be practical or cost-effective for semiconductor manufacturing. On the other hand, it wouldn't necessarily be required for all dopants or semiconductor materials.

Crommie said the purpose of his work was not to make devices but to conduct basic research on fundamental molecular behavior. "We hope to eventually create devices at a molecular scale, but if we want to create new devices, we need to understand small molecular structures."

"There is a great deal of interest with no practical applications as of yet," said Jun Nogami, University of Toronto professor of materials science and engineering.

It could be years before practical devices are made commercially using this technology, Nogami noted. ■

—Linda Dailey Paulson

PARC Develops Software to Connect Devices

Palo Alto Research Center scientists have created a technology that promises to let consumer-electronics devices communicate with one another and access content and resources across hardware, software, and networking platforms.

PARC's interoperability technology, Obje, is designed to make it easier to use a single device to access many dif-

ferent types of content and resources. And the software would help providers save money by letting them design just one version of their content for multiple platforms.

Industry observers say this could increase the adoption of digital media and the devices that play it. Adoption has been slower than hoped for, largely because of compatibility issues.

The Obje software architecture establishes a common means of communication to provide interoperability across platforms, said Hermann Calabria, PARC's principal of business development.

In a sending device, Obje recognizes the capabilities that the receiving device needs to work with code that it is transmitting, such as a codec to work with MPEG files. It also recognizes which of those capabilities the receiving device lacks. Obje then sends the recipient the code that will provide the missing capabilities. An older machine that can't work with Obje can work via a proxy device.

With Java-enabled devices, Obje can use a Java virtual machine, rather than send missing code, to achieve device, OS, and network neutrality. A JVM interprets compiled Java binary code so that a processor can perform a program's instructions. Any Java program can run on a platform for which a JVM has been designed, thereby enabling platform interoperability. Calabria noted that Obje could also work with cross-platform approaches other than Java.

The technology need not be preloaded on every networked machine to work, as long as machines can connect to an Obje-enabled device—such as a set-top box or stereo receiver—that could serve as a hub.

Researchers hope to make Obje usable on handheld devices. So far, they have shown that the software will run on resource-constrained devices by testing it on a Hewlett-Packard iPaq PDA. They are planning more tests, although Obje currently is too expensive and requires too much memory and processing power for handheld devices.

Even if Obje proves to be technically successful, the technology's marketplace success will still depend on industrywide adoption and promotion. PARC sources say they have spoken with several companies, which they declined to name, about turning Obje into a commercial project.

Home-entertainment networking, which Obje would facilitate, is a hot topic because of its enormous profit potential for device makers and content providers.

Until now, said Vamsi Sistla, director of broadband and residential entertainment technologies for ABI Research, a market analysis firm, "No one technology or standard or protocol has been able to address the many networking and technological challenges. There are already more than 60 or so standards and protocols."

To address compatibility issues, companies have formed and joined industry groups such as the Digital Home Working Group (www.dhwg.org/home) and the UPnP (universal plug and play) Forum (www.upnp.org). ■

—Linda Dailey Paulson

Creating Blazing Hot Images of Fire

When moviemakers need to show fire, they traditionally use the real thing because animations typically haven't had the kind of detail that makes the images effective. However, using real fire can create safety hazards and limit the size of the blazes that can be shown.

Now, though, Stanford University assistant professor Ron Fedkiw; University of California, San Diego, assistant professor Henrik Jensen; and former Stanford postdoctoral student Duc Nguyen have developed software to create realistic fire animations, overcoming shortcomings that have kept moviemakers from using such applications in the past.

Previously, generating realistic computer simulations of fire was difficult because it required a great deal of detail, Fedkiw explained.

With the new software, users can set initial conditions for the fire, such as temperature, type of fuel, and surface shape. Taking advantage of today's powerful computers with large amounts of memory, the technology solves equations that describe swirling fluids, expanding gases, and vaporized fuel, then it renders images such as smoke, soot, and igniting objects.

The software uses this information to create animated flames in much the same way that fire actually occurs. The technology has elements combust at different times, in different patterns, and in different colors depending on the heat level, and it creates images of black soot and smoke as they cool over time, as would happen in a real blaze. The software also adds many fine details, such as eddies in swirling smoke, to make the images more effective

The technology takes about five minutes to generate a frame of animated fire, Fedkiw explained, which is comparable to the time it takes to render many photorealistic animations such as those of hair and water.

He said enabling greater and easier control of fire animations could require another year or two.

Filmmakers and special-effects companies have expressed interest in the fire-animation software, according to Fedkiw. The technology could also be used for applications such as virtual reality training for firefighters. ■

—Linda Dailey Paulson

System Uses Existing Attacks to Predict Future Threats

A US company has developed a technology for analyzing current computer intrusions and extrapolating them to determine how future assaults might look, even before hackers develop them.

The ability of Icosystem's technology to identify at least some new types of attacks could overcome the limitations of many firewalls, antivirus programs, and other security applications that recognize only the code signatures or attack patterns of known assaults.

Eric Bonabeau, Icosystem's chair and chief scientific officer, said that while predicting new attacks would be a beneficial side effect, his technology's main goal is to discover systems' security vulnerabilities.

Today's security systems typically analyze traffic for signs of past malicious activity, such as specific code strings or data arriving at an unusual TCP/IP input port. Icosystem's technology takes information from known intrusion and virus approaches, sometimes found in online hacking software, and uses artificial evolution to show how the attack scripts might morph.

The system systematically changes the scripts to find the most deadly mutations. The Icosystem technology also combines parts of hacker programs to see how new attacks would evolve.

The security software could then implement predesigned defenses for vulnerabilities to these most likely assaults or recognize the signs of hackers launching these attacks, via code strings, entry ports, or other signatures.

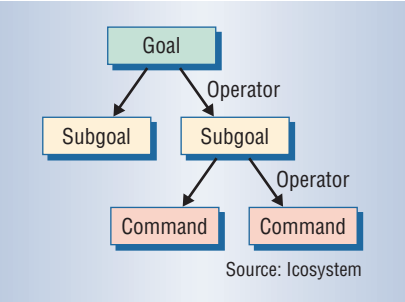
The US Army's Computer Crime Investigation Unit sponsored the experimental phase of the technology, which is still in the prototype stage. "There are lots of details to be worked out," Bonabeau explained.

Some industry observers say the approach could lead to a smarter generation of intrusion-detection systems. But Eric Ogren, senior analyst for the Yankee Group, a market research firm, expressed skepticism.

Ogren said Icosystem's technology might be a good system-auditing tool to identify vulnerabilities but would be minimally effective as a safety tool because it would only detect, not prevent, attacks. Also, Icosystem's technique wouldn't help with new attacks that aren't based on old approaches.

Bonabeau said Icosystem may never release a commercial version of the system because it might require more work and staff expertise than companies have available for such purposes. ■

—Linda Dailey Paulson



Icosystem has developed a technology for analyzing computer intrusions and extrapolating them to determine how future attacks might look. The system does this in part by abstracting a hacker's goals from the commands actually used in an attack, via subgoals in between, and then generating scripts of potential new assaults based on the initial incident.

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

Who Is Liable for Insecure Systems?

Nancy R. Mead
Software Engineering Institute

A survey of security-related liability issues shows that as data crime increases and the costs of hacking and other malicious behavior soar, vendors must make their software more secure or suffer market and legal consequences.

The notion of liability for insecure computer systems has shifted. Over the past several years, we have seen both the enactment of legislation affecting liability and the appearance of actual liability cases in the courts. For many years, computing professionals have debated the topic of liability for insecure systems. Erin Kenneally[1] provides a summary of this debate's legal aspects and observes that claims against insecure systems have evolved from a veiled threat into a ripening promise.

Several software flaws can make systems insecure. According to Kenneally, they derive from the current software status quo:

- Vendors are aware of errors and vulnerabilities in software created by their coding.
- Vendors can foresee the potential for mass failures and disruptions due to exploitation of insecure software.
- Users know that misconfigurations and software flaws can allow an intruder to gain unauthorized access to their systems.
- Users know that vendors provide patches, upgrades, and notices to address software vulnerabilities.
- Users have no knowledge of insecure design and no ability to alleviate its effects.
- Onset time for exploiting vulnerabilities starts as soon as the product is installed.

Kenneally concludes that it is reasonable for users to assume that software products will not allow intrusions. She further notes, however, that software products have, to date, had security holes that allowed intrusions costing billions of dollars.

Because of these flaws, an entire industry has developed to try to mitigate security holes after the fact. Software vendors develop and announce patches for security holes. System administrators busy themselves applying security patches. Other vendors sell products such as virus detection tools, and consultants perform penetration testing. Network technicians install firewalls and configure them to block intrusions. Clearinghouses provide a continual stream of information about the latest viruses, worms, and Trojan horses. The issue of liability lurks just beneath the surface of all these activities.

0018-9162/04/$20.00 © 2004 IEEE

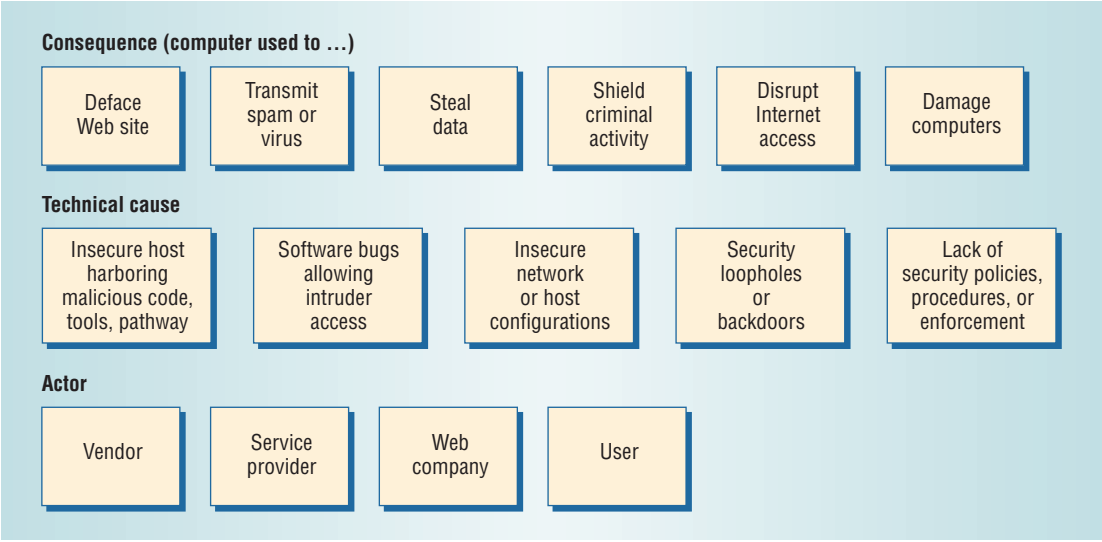


Figure 1. Liability test worksheet. System administrators can use the worksheet as a template to help identify potential attacks and the associated liability.

LIABILITY OVERVIEW

Software and data quality share an important relationship. As Joseph Juran observed, "Data are of high quality if they are fit for their intended uses by customers in operations, decision making, and planning."[2] Recently, we have seen an emphasis on accountability for data and software quality. Specifically examining data, we are told that those who create data models and values must be held accountable for the quality of those models and values. Likewise, those who develop applications must be held accountable for the quality of the data's presentation.[2]

Sources of corruption

In a related discussion about corrupted data, Robert W. Pautke suggests eliminating the sources of corruption and applying human and financial resources to protecting only the most important data.[3]

Corrupted data is only one piece of the puzzle, however. The suggestion here is that hackers disrupt not only networks, but also applications and data. Indeed, many researchers feel that if we can protect applications, it might not be necessary to defend networks so rigorously. In an editorial commenting on software quality,[4] Karl Reed describes a litany of all too familiar software woes and cites massive "societal costs due to wasted time."

The notion of accountability for software and data quality naturally leads to the question of manufacturers' liability for software. Several years ago, a study that used students as attackers to evaluate commercial off-the-shelf (COTS) software products yielded the following results:[5]

- Most attackers performed successful intrusions.
- Several of the intrusions gave the attacker administrator privileges.
- The Internet provides a vast amount of information on how to successfully attack common systems.
- Many attackers broke into the system by using exploit scripts published on the Internet.

Liability concerns related to security fall into several categories. They include concerns about safety-critical systems, Internet-connected consumer appliances, businesses with vulnerable computer networks, distributed denial of service (DDoS) attacks, and vendors who produce commercial software.

Kenneally provides a tool to help determine potential liabilities.[1] System and network administrators can use the liability test worksheet shown in Figure 1 to identify the kind of agents who perpetrate data crimes, the means used to compromise target systems, and the specific nature of these attacks. Figure 2 shows a sample worksheet that links various causes, agents, and data crimes.

Determining liability

Some computing experts have argued that manufacturers and suppliers of computers associated with safety-critical systems should be liable for injury those systems cause, including injury that results indirectly from security holes.[6] This suggests that liability resulting from security problems in Internet-connected appliances could be treated similarly to general-appliance liability. If appliance manufacturers are liable for other types of failure, they or their vendors could also be held liable for failure resulting from security holes.

Many analysts recommend that the burden of protection against attack should lie with companies and businesses that have the resources to provide such protection, not with the general public. They discuss liability for monetary loss as a result of hacking, as well as reducing liability by the appropriate use of contracts and insurance. Another report suggests that firms most capable of protecting against DDoS attacks are those most

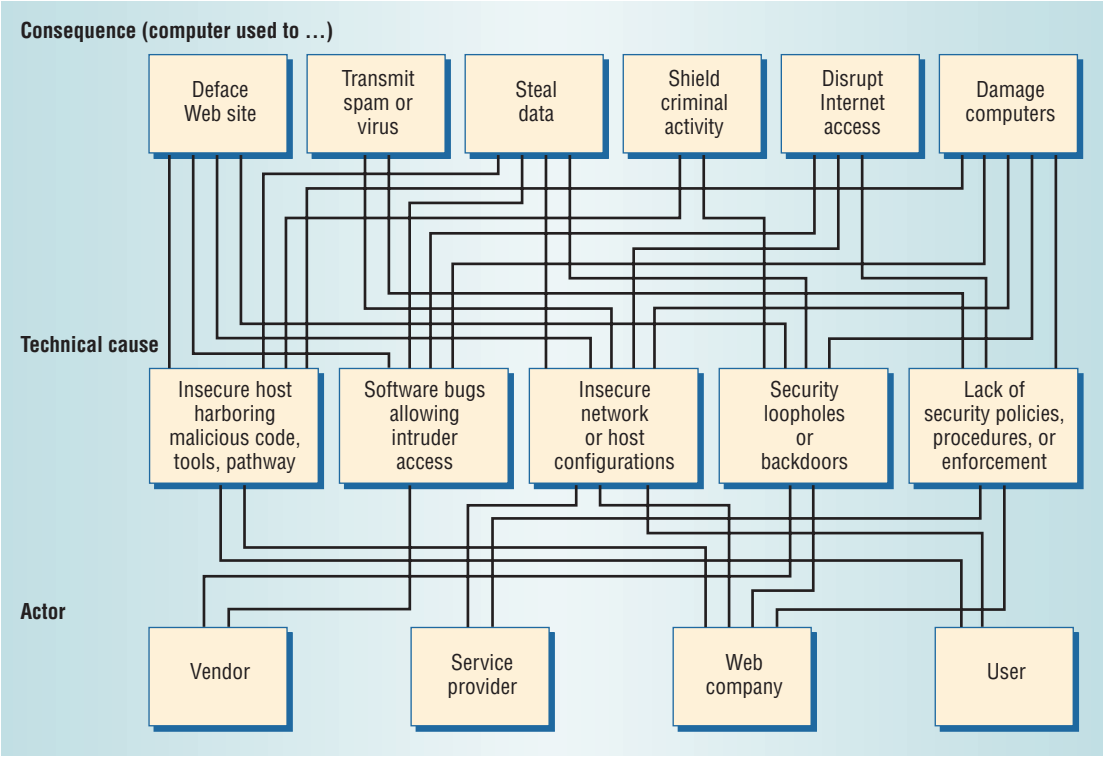


Figure 2. Potential liabilities. These are filled-in examples of potential intrusions and associated consequences.

likely to be liable if damage occurs.[7] The National Academy of Sciences recommends that software companies be held liable in cases in which buggy or insecure software causes security gaps.

Another view suggests that security is both an economic and technical issue. Finding all the security holes in a particular piece of software is expensive, but this must be done to ensure security. Yet it costs a hacker little to find a single security hole to exploit. So the numbers tend to work against the security effort. An emphasis on making software easy to use provides another disincentive that tends to mitigate against increased security.

Some companies demand liability clauses in contracts with vendors to hold them responsible for any security breach connected to their software. Officials at these companies expect such language to become more prevalent in contracts. They see the trend as a constantly reinforced motivator urging programmers to focus on security. They also recommend that companies protect against liability suits by adopting policies that adhere to information security standards and best practices.[8]

Diverse viewpoints

A recent series of articles by two software liability experts shows just how divergent individual viewpoints can be. On the one hand, some experts argue that expecting consumers to create their own security software is impractical.[9] They assume, rather, that it is only reasonable for software manufacturers to ensure their products' reliability. The notion of *fitness for use* suggests that such software provides a basis for *strict liability*, a concept that has been applied in personal injury cases, but not so much in property damage cases and even less in economic damage cases. In these latter cases, the courts often accept contractual disclaimers of liability, such as those found on shrink-wrapped software.

Network security and the associated liability issues have also received attention. A negligence argument can be made in support of liability for insecure networks. This argument would need to show that the insecure party "had a duty to use reasonable care in securing its computer systems, breached that duty by failing to employ adequate security, and was a reasonably recognized cause of actual damages."[10]

Since strict liability is not applicable in all cases, the courts must build on other concepts, such as "warranty of fitness, misrepresentation, abnormal danger, negligence, fraud, lack of clarity, and unconscionability to find liability for all security product failures."[9] The doctrine of unconscionability has been applied to contracts and, when safety is at issue, disclaimers have been invalidated. This argument suggests that litigants can successfully make liability arguments in the US legal system for at least one class of software products.

The opposing view suggests that liability is not the appropriate tool for reducing the number and severity of software security holes.[11] This argument suggests that software does not have the characteristics of other physical products that would support legal liability. For one thing, a liability case could take years to move through the courts, in which case the software—having a relatively short

Users demand flexibility, but this makes a given software product's uses less predictable.

life cycle—could become obsolete before the case came to trial. Thus, even if software liability is confirmed, the outcome could be irrelevant.

Another argument against software liability holds that manufacturers cannot predict how software will be used or where and how it will be installed. This lack of predictability makes it impossible for manufacturers to warrant software for fitness of use. For example, users can tailor many commercial software products through preferences. Users demand this flexibility, yet it makes a given software product's uses less predictable, thus making it secure is more difficult.

In the past, the courts have distinguished between manufacturing defects and design defects, with the notion of software liability typically resting on design defects. A software manufacturing defect could occur, for example, if an improperly produced software CD made the product either impossible to install or otherwise invalid. Liability actions would seldom take place in such a case, because the standard disclaimers usually provide for only replacing the product or refunding the purchase price.

The final argument that liability is inappropriate for software suggests that many software manufacturers would leave the business if faced with liability suits, leaving few remaining manufacturers to service a large and burgeoning marketplace.

Given the diversity of views regarding how to assign liability when a system is compromised, it's not easy to identify a uniform approach to solving the problem.

UCITA

The National Conference of Commissioners on Uniform State Law developed the Uniform Computer Information Transactions Act from a series of more than 20 three-day meetings by the Drafting Committee.[12] As many as 120 interested observers attended each session. Now in its 112th year, the NCCUSL comprises more than 300 lawyers, judges, and law professors appointed by the US's 50 states, the District of Columbia, Puerto Rico, and the US Virgin Islands. Its members draft proposals "for uniform and model laws on subjects where uniformity is desirable and practicable, and work toward their enactment in legislatures."[13]

The NCCUSL approved UCITA, a uniform statute intended to codify current law and practice in contracts for computer information. UCITA does not purport to answer every question, but it does provide a framework within which courts can analyze questions. The rationale for UCITA's development stems from the combination of growth in the information economy and the corresponding lack of a uniform framework for licensing. UCITA sets up a series of default rules that apply in the absence of a specific agreement by the interested parties.

Virginia adopted UCITA first, after a one-year study period by a special legislative committee. Although Virginia's legislature considered many amendments, it ultimately adopted UCITA in 2000 with no significant change. Maryland also adopted the act in 2000.

UCITA's scope is limited to transactions in computer information. Such a transaction constitutes "an agreement or the performance of it to create, modify, transfer, or license computer information or informational rights in computer information." UCITA defines computer information as information in electronic form, obtained from or through the use of a computer, or which is in a form capable of being processed by a computer. Under UCITA, contracts can be made computer to computer or human to computer. Further, the act codifies existing case law for shrink-wrap and click-wrap contracts.

Section 105 states that federal law preempts UCITA, as do state consumer protection laws. Contracts under the act may not contain terms that violate fundamental public policy. For shrink-wrap contracts, a licensee may not *manifest assent* to the terms of a license until he or she has had an opportunity to review them. If the license is presented after payment, the license must provide the licensee with a cost-free right of return.

For click-wrap contracts, similar rules prevail. The licensee must have the opportunity to review the terms prior to manifesting assent. Further, UCITA encourages pretransaction disclosure of terms in Internet transactions:

> You 'manifest assent' if, after having an opportunity to review a record or term, you authenticate the record or term, or intentionally engage in conduct or make statements with reason to know that the other party or its electronic agent may infer from the conduct or statement that you assent to the record or term. … A person has an opportunity to review a record or term only if it is made available in a manner that ought to call it to the attention of a reasonable person and permit review.[12]

UCITA creates, for the first time, statutory implied warranties in information transactions. The warranties include noninterference and noninfringe-

ment, merchantability of the computer program, informational content, fitness for the licensee's purpose, and system integration.

Many computing professionals perceived that the UCITA framework provided liability protection to vendors. Those involved in developing UCITA, on the other hand, argue that all UCITA does is transfer existing legal results from one medium to another. Many responsible professional software organizations and engineers feel that UCITA goes too far in protecting vendors from liability. These issues continue to generate considerable discussion.

The UCITA authors contend that consumer advocates sought broad consumer protections within UCITA, rather than leaving it to individual states to develop them—a precedent that arises from traditional consumer law. The original UCITA positions on reverse engineering, public comment, and electronic self-help were opposed by computing professionals, as were the original positions on default rules for number of users and duration of license. Some computing professionals oppose shrink-wrap contracts on principle. The NCCUSL never responded to this last point.

In 2001, the American Bar Association appointed a special working group to evaluate UCITA. After a three-day meeting held to discuss these concerns, the ABA issued a report suggesting 11 specific and substantive changes, including the following:[14]

- *Electronic self-help banned.* Vendors of digital information, including software, may not disable the use of that information by electronic means if an information contract is breached. Vendors have an expedited remedy for a material breach of contract in a court of law.
- *A state's consumer protection law trumps UCITA.* An information contract is expressly subject to and cannot waive any consumer protection provided in state or federal law. Included are laws providing for conspicuous disclosure, unfair or deceptive trade practice laws, and laws relating to electronic signatures and records.
- *Right to criticize protected.* Information contract terms that prohibit criticism of an information product are unenforceable. Parties can contract in a manner consistent with other law, such as the law of trade secrets.
- *Remedies for known material defect preserved.* Such remedies are expressly made available for information products as fully as for defective goods or services.
- *Reverse engineering for interoperability expressly authorized.* An information contract cannot prohibit reverse engineering done for the purpose of making an information product work together with other information products.
- *Special open source software provisions.* The act expressly excludes coverage of open source software if only copyright permission is given and is not part of a contract. If there is a contract, there are no implied warranties if the transaction generates no commercial gain.

Focusing only on features tends to result in buggy, insecure software.

NCCUSL also adopted 38 amendments to improve clarity, with no substantive effect. The ABA House of Delegates was to consider UCITA approval in February 2003. The ABA, however, preferred not to take a position on UCITA when it became clear that a consensus was unlikely to emerge.

UCITA was under consideration in additional states. The American Electronics Association, a high-tech trade association with more than 3,000 members, endorsed UCITA, but software professional organizations and individual professionals continued to express concern.

The NCCUSL has dropped active lobbying for adoption of UCITA.[15] To date, only Virginia and Maryland have actually adopted UCITA. Although another five states were expected to pass the proposal last year after amendments were made to address consumer concerns, this did not occur. The amendments did not satisfy the American Library Association, the American Bar Association, or the Americans for Fair Electronic Commerce Transactions. The NCCUSL is not expected to revisit UCITA for another four to five years. At present, it seems likely that if litigation occurred, a vendor would try to benefit from having the case heard in Virginia or Maryland.

Lacking a uniform code that specifically describes the level of system security computer users can reasonably expect, the average user must look elsewhere, for example, to software vendors for improved security.

COTS DEVELOPMENT PROCESSES

Most COTS vendors focus on providing features. They typically eliminate the most serious problems to be first to market, then fix other problems after release. This focus on features tends to result in buggy, insecure software because, until recently, security has not been a vendor priority. Major vendors—including IBM, Microsoft, and Sun—have

Major vendors have produced tools to aid in the development and debug process.

produced tools to aid in the development and debug process and to eliminate bugs that could lead to security holes.

Microsoft security push

Microsoft has made a major commitment to security improvements in its software, spurred on by a now-famous memo from Bill Gates (www.computerbytesman.com/security/billsmemo.htm). This memo, drafted in January 2002, resulted in significant changes within Microsoft.[16] For one thing, during February and March 2002, all Windows feature development stopped while the Microsoft team analyzed design, code, test plans, and documentation. For a market-driven company, this is an unusual step, and it shows just how seriously Microsoft took this security initiative.

Training courses were developed and delivered to support the Windows Security Push. As part of this effort, the team observed that security is not just a layer added after the fact, but a consideration that pervades all of development.

A key element of the design process was the construction of threat models. In this regard, there is some parallelism between the Microsoft effort and the guidance of the Common Criteria. Microsoft uses the following steps when constructing threat models:

1. Decompose the application to determine the system's boundaries or scope.
2. Determine threat targets and categories using components from the decomposition process and determine the threat categories for each target.
3. Identify attack mechanisms using a threat or attack tree approach.
4. Respond to the threats with mitigation techniques appropriate to each threat.

Obviously, this was and continues to be a massive effort for Microsoft, which stated that the need for more secure platforms to support future solutions, a prior successful effort to push .NET, and the ability to respond to a new generation of Internet threats prompted this push. Further, the company expects that customers will perceive improved security in the products.

I wonder, however, whether a concern about possible future liability is also part of Microsoft's agenda. Regardless of the motives, this effort seems to have great potential benefits for consumers of commercial software products. Yet a recently published report[17] takes an opposing view and suggests that Microsoft has engaged in monopolistic practices that have in part contributed to excess complexity and security flaws. The report further observes that to improve security we must first address the dominance of Microsoft's products.

Liability litigation against Microsoft

In a recent California development,[18] a consumer claiming to be a victim of identity theft via computer hacking held Microsoft responsible for failure to secure its software against viruses, worms, and other cyberattacks, and initiated a lawsuit. Whether this will evolve into a class action suit remains to be seen, but the outcome will be extremely interesting from a liability viewpoint, especially in light of the Security Breach Information Act enacted in California effective July 2003. This act requires that people and companies doing business in the state notify customers of suspected security breaches and provides for civil damages, including financial awards.

OTHER REMEDIES

Although applying the notion of due diligence to software security could help reduce the possibility of companies being found liable for security holes, developers and users could also try other approaches.

Reduce vulnerable features

Only a few sophisticated users are competent to exploit many vulnerable features in COTS software. Vendors could thus provide a basic software set that does not contain advanced features, such as macros, for use by the average end user. Developers who need more sophisticated features could purchase an alternative version. These users would probably have the know-how to patch their own systems.

More centralized services

If consumers used centralized services such as WebTV instead of personal computers, the number of targets for attack would decrease. Many users do not need the myriad features that PCs offer. They only use e-mail, browsers, and a few standard office applications.

I don't know how likely it is that users will opt for Web-based services, but securing a smaller number of professional systems seems less daunting than securing every single personal computer in use.

Licensing and certification

Licensing and certification of software engineers could be viewed as a form of due diligence on the part of companies developing software, especially

COTS products. However, the number of licensed software engineers in the US is small, probably in the double digits. Further, as of October 2003, the IEEE Computer Society has certified only 150 software engineers worldwide as software development professionals. Given this small number, we can't expect licensing or certification processes to produce enough software engineers who can develop secure software.

Certification processes do exist for security professionals, but they generally apply to systems administrators responsible for securing systems after the fact, rather than software engineers engaged in initial development. Nevertheless, for companies using commercial software, employing certified system administrators could potentially help reduce security holes.

Software degree programs

Universities offer many degree programs in software engineering, both graduate and undergraduate, along with several degree offerings in information security. The content of such programs could contribute to ensuring that the programs' graduates will develop and maintain software that has fewer bugs and better security measures.

Unfortunately, if they cover information security at all, software engineering and computer science degree programs often treat it as an elective topic or lump it with other quality attributes or ethics. The tragedy here is that the vast majority of security holes result from only a few coding errors, such as buffer overflow. Thus, in the long term software development requires attention to security throughout its life cycle, not just after a system is operational.

COTS risk assessment

Assessment of the security risks associated with COTS software could be considered a form of due diligence on the part of organizations acquiring COTS software.[19] Typically, red teams and intrusion detection tools help shore up the use of COTS, not risk assessment at the time of acquisition. For example, the @stake Hoover Project[20] examined 45 e-business applications and developed application security profiles for each of them. This study focused on applications for two reasons:

- application-level attacks can easily traverse most firewalls and
- that is where the money is.

The most secure applications in the study contained about one-fourth of the defects found in the least secure applications. This study could provide insights for other application software users and developers. Much more could be done to assess and mitigate security risks associated with application software in general, and COTS software specifically.

Employing certified system administrators could potentially help reduce security holes.

Software is like the automobile. Years ago, no one considered suing an automobile manufacturer over injuries resulting from an accident. There was no such thing as a recall. Then Ralph Nader so effectively raised awareness of automotive problems that liability for automobile manufacturers resulted.

A similar epiphany will take place in software. Rather than viewing computers and their associated software as a giant black box, companies and consumers will decide that software should, reasonably, be fit for use at some level. They will also think it reasonable for manufacturers to have shown due diligence.

Therefore, companies and software vendors will try to protect themselves, although what constitutes a reasonable level of due diligence has yet to be determined. Although the courts may not be the best venue for resolving this, liability cases will take place and an evolution of best practices will occur as well. It's unlikely that 100 percent security will ever be achieved, but significant improvement relative to the status quo likely will occur, with liability cases being just one catalyst. ■

References

1. E. Kenneally, "The Byte Stops Here: Duty and Liability for Negligent Internet Security," *Computer Security J.*, vol. 16, no. 4, Fall 2000, pp. 1-26.
2. T.C. Redman, "Opening Statement," *Cutter IT J.*, Jan. 2003, pp. 2-5.
3. R.W. Pautke, "To Clean or Not to Clean, That Is the Question," *Cutter IT J.*, Jan. 2003, pp. 11-12.
4. K. Reed, "Good Enough Is Not Good Enough," *IEEE Software*, Sept.-Oct. 2003, p. 109.
5. U. Lindqvist and E. Jonsson, "A Map of Security Risks Associated with Using COTS," *Computer*, June 1998, pp. 60-66.
6. D. Davis, "Legal Aspects of Safety Critical Systems," *Proc. 14th Int'l Conf. Computer Safety, Reliability and Security*, Springer, 1995, pp. 156-170.
7. M.J. Radin, "Distributed Denial of Service Attacks: Who Pays?" *Computer Economics Report*, Int'l Ed., Aug. 2001, pp. 12-15.

8. R. Weiler, "Decision Support: You Can't Outsource Liability for Security," *InformationWeek*, Aug. 2002; www.informationweek.com/story/IWK20020822S0003.
9. D.J. Ryan, "Two Views on Security Software Liability: Let the Legal System Decide," *IEEE Security & Privacy*, Jan.-Feb. 2003, pp. 70-72.
10. E. Kenneally, "Who's Liable for Insecure Networks?" *Computer*, June 2002, pp. 93-94.
11. C. Heckman, "Two Views on Security Software Liability: Using the Right Legal Tools," *IEEE Security & Privacy*, Jan.-Feb. 2003, pp. 73-75.
12. M.J. Dively, "The Uniform Computer Information Transactions Act," 2003; www.nccusl.org/nccusl/ucita/ucita/BusLawAdvisor.pdf.
13. The National Conference of Commissioners on Uniform State Laws, Nov. 2003, Uniform Law Commissioners; www.nccusl.org/.
14. The National Conference of Commissioners on Uniform State Laws, UCITA 2002 Revisions Memorandum and Chart; 23 Aug. 2003; www.nccusl.org/nccusl/ucita/UCITA_082602_MEMO_and_CHART.pdf.
15. L.D. Paulson, "Proponents Drop Support for Controversial Software Proposal," *Computer*, Oct. 2003, p. 18.
16. M. Howard and S. Lipner, "Inside the Windows Security Push," *IEEE Security & Privacy*, Jan.-Feb. 2003, pp. 57-61.
17. D. Geer et al., "CyberInsecurity: The Cost of Monopoly—How the Dominance of Microsoft's Products Poses a Risk to Security," 24 Sept. 2003; www.ccianet.org/papers/cyberinsecurity.pdf.
18. J. Krim, "Suit Holds Microsoft Responsible for Worm Holes," *Washington Post*, 3 Oct. 2003, p. E05.
19. H.F. Lipson, N.R. Mead, and A.P. Moore, "Can We Ever Build Survivable Systems from COTS Components?" tech. report CMU/SEI-2001-TN-030, Software Eng. Inst., Carnegie Mellon Univ., Dec. 2002; www.sei.cmu.edu/publications/documents/01.reports/01tn030.html.
20. D. Geer, K. Soo Hoo, and A. Jaquity, "Information Security: Why the Future Belongs to the Quants," *IEEE Security & Privacy*, July-Aug. 2003, pp. 24-32.

Nancy R. Mead *leads the CERT Survivable Systems Engineering team at the Software Engineering Institute, Pittsburgh. She is also a senior member of the SEI's technical staff in the Networked Systems Survivability Program and a faculty member in the Master of Software Engineering and Master of Information Systems Management programs at Carnegie Mellon University. Mead received a PhD in mathematics from the Polytechnic Institute of New York. She is a senior member of the IEEE and a member of the ACM. Contact her at nrm@sei.cmu.edu.*

Issues in High-Speed Internet Security

As the SQL Slammer worm proved, current high-speed security solutions are immature and ineffective. Protecting networks against such fast-moving threats requires a new paradigm that offers flexibility, high performance, and speed.

Peder Jungck
CloudShield Technologies

Simon S.Y. Shim
San Jose State University

With global Internet access, people and organizations can share information instantly. Unfortunately, such access also leaves the Internet vulnerable to malicious actions that can cripple computers, businesses, governments, and lives on a scale never before possible. Hackers, worms, and viruses can unleash attacks with electronic as well as physical, economic, and safety ramifications.

Over the past decade, the threat of computer worms and viruses has grown from a nuisance to perhaps the greatest obstacle to the growth and reliability of the Internet and large networks in general. Although systems and network management best practices can help prevent some current threats, they can't protect against the unknown. Further, updating every system for every known threat has simply become too difficult for administrators to manage or for governments, companies, and individuals to afford.

Trends in worm and virus delivery mechanisms and infection speed have also changed. Not long ago, a virus warning and the patch to vaccinate computers against it would appear days before the virus began spreading. Today, too often the first sign of a virus is that a part of the network goes down. Flash worms such as SQL Slammer have paved the way for future worms to carry payloads that directly target their victims and wreak havoc on government, business, and societal structures. Existing technologies such as firewalls, intrusion detection systems, intrusion protection systems, virtual private networks (VPNs), and virus scanners provide integrated security solutions. [1–4]

Not surprisingly, security has become a massive industry, and it is now a focal point for virtually every organization. Proactively eliminating just the known threats places an impractical burden on existing server and network infrastructures. Eliminating unknown threats or *day zero* attacks—which, as the name implies, reveal themselves only when they first occur—requires new real-time solutions that can identify unique attacks without overburdening the network with security and management overhead. To address both the threats facing networks today and future scalability demands, we need new security methodologies, deployment strategies, systems, and architectures.

SQL SLAMMER

The SQL Slammer, one of the first flash worms unleashed on the Internet, took down networks around the world in January 2003.[5] Flash worms spread in an instant, doing damage almost before administrators can react and adapt defenses.

The speed with which SQL Slammer spread and the damage it inflicted imply rampant network insecurity. It took just one 384-byte packet penetrating a single network to set off the carnage. The worm needed no session or conversation start-up sequences. The packet, a subset of which appears in Figure 1, exposed no fragmentation issues or other common signatures. Many firewalls left the port it attacked open to provide a service, and most intrusion detection systems left that port unmonitored.

Published by the IEEE Computer Society
0018-9162/04/$20.00 © 2004 IEEE

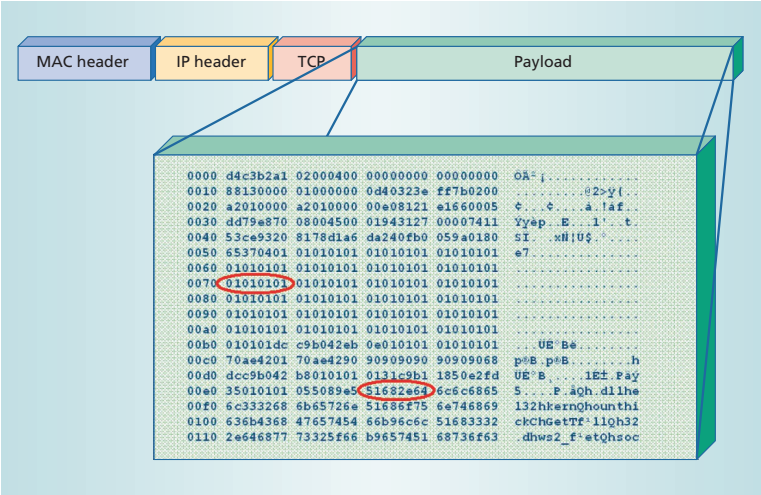


Figure 1. SQL Slammer worm packet subset. The circled portions show two common traits of many network-borne worms. The 01010101 repeating pattern is padded data used to consume an input field on the recipient system, creating a buffer overflow. The second circled portion is the actual code intended to execute on the exploited system. In identifying worms, signature-based systems look for buffer exploits in general, such as repeating 01010101, and at times identify the signature of the worm's code base through unique code patterns found subsequently in the packets.

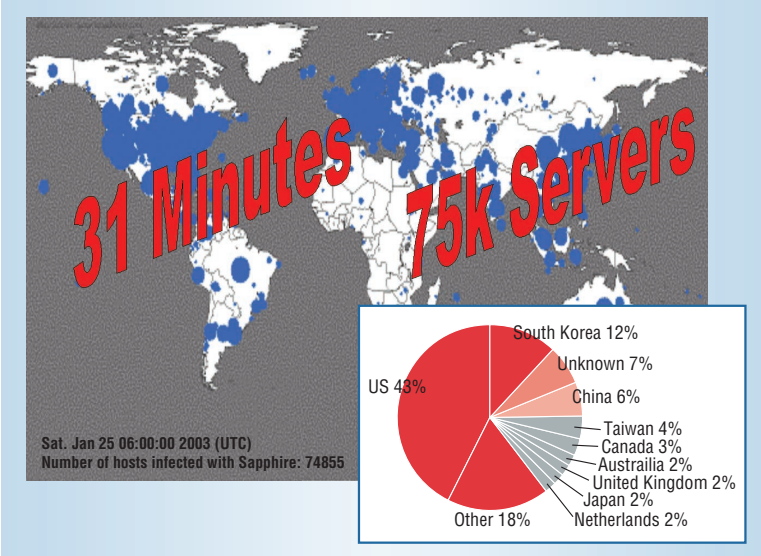


Figure 2. SQL Slammer worm attack. On 25 January 2003, the worm infected more than 75,000 servers around the world in a mere 31 minutes.

Fortunately, the worm merely replicated itself instead of more aggressively damaging the networks. Nevertheless, the worm's impact changed how we view network security.

Because it was small and used a single infected system as a base from which to infect other networks, and because so many SQL servers were connected to the Internet, SQL Slammer crawled around the globe in minutes. Unlike previous widespread virus or worm attacks, the worm's spread wreaked as much havoc by overloading the infrastructure as it did by actually taking down networks. In 10 minutes, it had spread throughout the world; in 31 minutes, more than 75,000 systems were infected, as Figure 2 shows. This all occurred long before most administrators had even taken a first step toward figuring out what was going on.

After the first infection, the SQL Slammer used the same packet format to launch additional attacks, causing some systems to generate as many as 30,000 packets per second. Each newly infected server targeted a random distribution of other IP addresses to infect. The available tools were limited or resembled a sledgehammer in their ability to defend against the worm: Filtering access-control lists (ACLs) might lock out valid SQL communications, and antivirus and intrusion protection systems provided little protection because they didn't know the signature to look for. Although rebooting removed the worm, the servers were still susceptible to reinfection and manually initiated downtime after each event. SQL Slammer went straight through the holes that existing security devices left. Worse, discerning the "single-packet killer" required performance capabilities that few systems possessed.

Given that the Internet is a distributed network that spans the globe and that day zero worms do exist, similar strikes are unavoidable. However, identifying new threats and locking down networks when these threats first appear can largely mitigate their impact. Combining these capabilities with automated network self-healing technologies lets rollbacks return the network to a preattack state. As networks become self-aware, rule sets become self-adapting. As collaboration develops between autonomous network elements—not just centralized administrators and their systems—addressing threats at the speed they spread may become possible.

Until developers create new breeds of systems and software, however, countering threats will remain a time-consuming, firefighting, no-win situation. Point solutions, which reflect defensive and proactive measures, continue to be cost inefficient, complex to manage, and difficult to scale and adapt to new threats. If a system allows access from outside networks, router ACLs will probably let the attack in. By allowing communication between at least one peer SQL server and a computer system, a firewall could leave the window open for infection. Allowing systems to initiate communications from inside or outside VPN-peered networks offers another infection mechanism.

Herein lies the problem: Existing network security products can verify that trusted parties com-

municate, but what occurs when a fast-spreading worm infects a trusted party? Unfortunately, in the case of SQL Slammer, too many enterprises found out too late.

Technology won't evolve fast enough to satisfy the current market demand for better network security.

NETWORK SECURITY AT SPEED

Most network managers will willingly sacrifice speed for security, or vice versa, depending on the user community's priorities. Fast and safe networking has either been technically impossible or too costly for most companies. As the speed of threats increases, consumers, businesses, and governments all demand better network security. The dilemma is that technology won't evolve fast enough to satisfy the current market demand for better network security.

Bandwidth versus processor speed

Why can't networks be both fast and secure? Today's optical-speed networks can't be made more secure primarily because bandwidth power has exceeded microprocessor power. That is, servers built around microprocessors and traditional PC architectures can't function at the speed of the fiber bandwidth already deployed. Closing this gap between bandwidth and processor speeds requires slowing traffic to a point at which servers or network devices can perform security applications to the data, or significantly curtailing or dropping security applications to meet network performance goals.

ACLs are fairly simple router-based security applications that illustrate the processor gap problem. When turned on, ACLs dramatically increase router CPU use, which significantly degrades router-forwarding performance. Network managers therefore must choose between turning ACLs off entirely, thereby putting their networks at risk, or buying two routers—one to route traffic and the other to run ACLs. Neither option is acceptable. Even multiple devices might not increase performance enough to keep up with the potential traffic deluge.

Although service providers obviously want to leverage their investments in existing routers and switches, redesigning such single-purpose devices to perform another function—such as providing security—usually requires sacrificing speed, cost, performance, or some other aspect. Similarly, devices designed to run at megabit speeds cannot be overhauled to run at light speed. Consequently, network architects can design optical network security solutions that either run slower with current equipment or faster with fewer applications, but neither alternative adequately protects the network.

Device flexibility

Flexibility is another important issue. Many high-speed security devices trade performance for flexibility. By predefining packet header fields or limiting operations against the packet's payload, developers can make assumptions that improve performance.

The Internet, the protocols traversing it, and the threats against the protocols have changed significantly in the past 10 years. Where once the definition of a header involved only legacy layers 2, 3, or 4 of the Open Systems Interconnect (OSI) model, it now also requires analysis of layer 7. Understanding a network conversation's context requires interpreting facets of layer 7, often referred to as the layer-4 packet header payload.

Deep packet inspection lets network devices peek into a packet's payload for certain limited-functionality data sets. Often this functionality provides the ability to perform a string search against the contents of text-based protocols or a binary pattern match against nontext-based protocols. *Full packet inspection* involves fully interrogating the additional application layer headers and making correlative analysis of the data as a structured data type in the payload.

The SQL Slammer's rapid spread occurred because it had a unique signature that only systems capable of performing deep or full packet inspection could identify. Few of the deployed deep packet inspection devices could easily reconfigure and filter the backbones, enterprise gateways, and local networks within service providers and enterprises.

Although some fairly simple payload-analysis techniques can identify SQL Slammer, new defensive solutions must be prepared to counter the human cunning of malicious developers with signature, anomaly, behavioral, and policy-management capabilities that administrators or automated heuristic engines can tune at a moment's notice.

ALTERNATIVE TECHNOLOGIES

The demands of optical-speed network security easily outstrip the ability of current network security products to provide 100-percent packet inspection with no degradation in network performance. Simply put, network security has become a bottleneck. Most network topologies are complex systems of point-product solutions tied together with load-balancing devices and application-integration middleware—the result of network managers demanding best-of-breed products in an integrated network as well as the network security industry's focus on individual network security solutions.

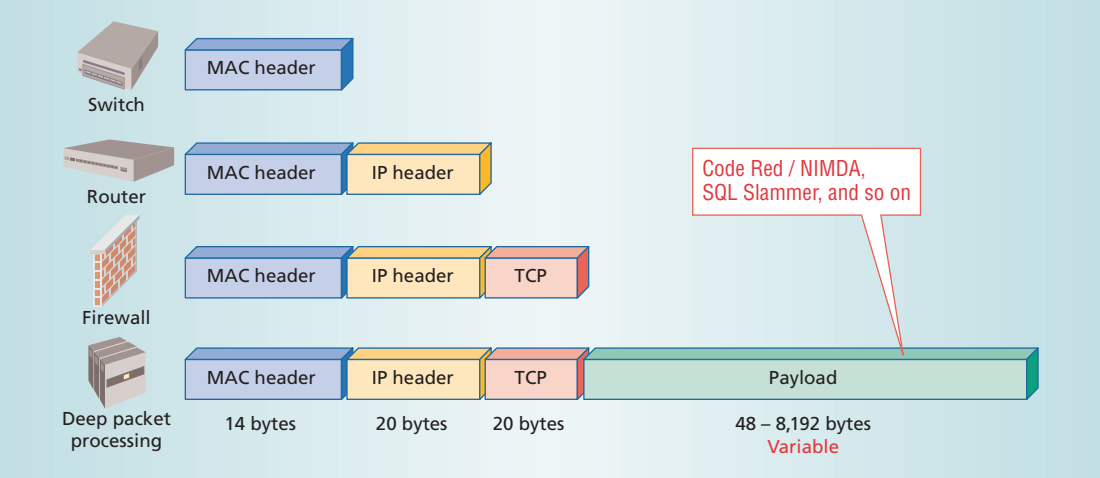


Figure 3. Packet inspection of core category functionality. Few devices offer deep packet inspection, which is necessary for early detection of attacks such as the SQL Slammer worm.

Many current network security devices address only certain layers of the network OSI stack. Virtually none offers the new security benchmark: 100 percent packet inspection of every packet at every network layer as the packets arrive on the network interface. Further, ownership costs, the blurring of lines between security devices, and the onslaught of other issues, such as traffic management and regulatory compliance, call for the convergence of capabilities into singular platforms.

Older security solutions are inadequate for processing security applications at light speed. Although network providers can transmit data from point to point very quickly, the processor gap makes it impossible to perform meaningful applications on that data without substantially slowing the network. Unless network providers can process security applications at line speed—often multigigabit data rates—the data traversing their high-speed infrastructure and the enterprises they serve are vulnerable to attack.

Although each class of network devices has benefits, none offer the combined speed, flexibility, scalability, and performance required to perform security applications on today's optical-speed network. Each legacy product class has at least one significant drawback with regard to optical-speed network security. Four categories of devices illustrate this point. Figure 3 shows the levels of inspection performed by these devices.

Routers and switches

Routers and switches inspect packet headers and redirect data flow to output ports on the device. Designers have optimized Internet protocols and routing algorithms to allow scalable application-specific integrated circuit (ASIC) implementations that maintain multigigabit data rates. Switching fabric technologies now support the demands of bandwidth scalability, which began increasing significantly in the late 1990s.

This performance and scalability comes at the cost of reduced flexibility. Some current ASIC implementations have fixed constraints around the packet format and processing algorithms. As security, quality of service, and other traffic management functionality requirements have increased, designers have often relegated exception handling to traditional processors.

Unfortunately, dealing with modern threats requires far more than the functionality and performance that "by exception only" provides. The exception has become the norm, and the 12- to 18-month ASIC line card turn time cannot compete with the daily morphing of threats.

Firewalls

Firewalls and other traditional security gateways delve deeper into packets and continue to improve line-rate processing.[3] Assuming burdens beyond traditional network ACLs, stateful filtering and application proxies maintain information about each connection and its state. Although this is a dramatic advancement over a router's stateless analysis, it still doesn't allow the flexibility to look deeper into the packet at fields not originally engineered for analysis.

Detecting modern threats such as worms and rapidly spreading viruses takes more than analyzing a few additional fields in a packet header or scanning a portion of the payload. Even so, this broad category of solutions continues to grow, with options ranging from simple software to incredibly powerful silicon. As with routers, although some gateway benefits might diminish over time, their importance remains critical to the infrastructure.

Servers

Servers are experiencing a great migration of functionality and architecture. Many network infrastructure devices—notably routers and firewalls—began as servers. Many security devices, whether marketed as software or appliances, are off-the-shelf or accelerated servers underneath. Blade servers converge processing with switching (layers 2, 3, and 7), and network interface vendors have driven the

Table 1. Technologies for addressing high-speed network security.

Technology	Description
Network processor unit (NPU)	Basic field comparison and analysis, coordination with coprocessors, control plane coordination
Field-programmable gate array (FPGA)	Packet scan inline functions (CheckSum validation), bus transformation and manipulation, multidevice coordination, custom coprocessors (decanonicalization)
Ternary content addressable memory (T-CAM)/Network search engine (NSE)	Large wire-speed silicon databases (access control lists/network address translation), packet-flow state data
Classifier	Payload string searches, variable size comparisons (URL/DNS), range-based analysis

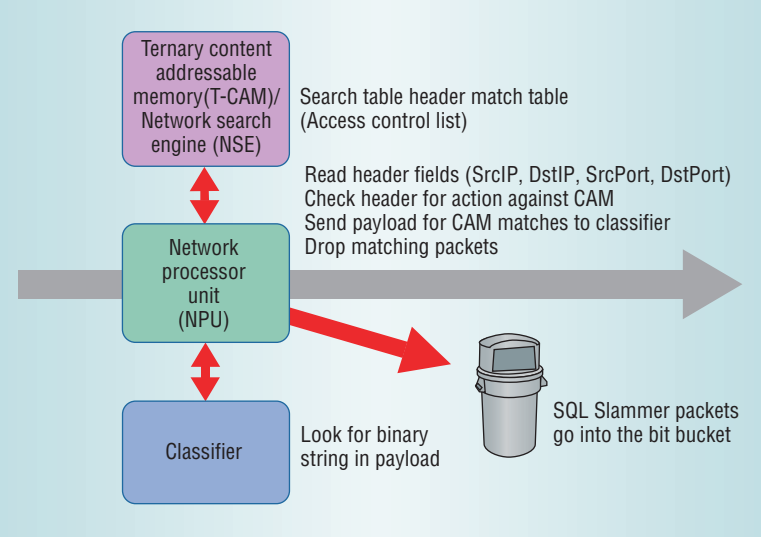


Figure 4. Simplified SQL Slammer subsystem approach. The processor interacts with the network search engine and classification coprocessor to identify the malicious packet without preventing valid packets from accessing the system.

advance of TCP offload engines. Advancements in both blade servers and TOEs continue to push the PC beyond Moore's law, attempting to satisfy not only computation but also data rate performance goals. Advancements notwithstanding, the performance achieved and the inherent issues of a PC architecture in the network infrastructure space will continue to hamper long-term success.

TOE developments have greatly benefited applications like Web servers, which provide simplified requests that are computationally and storage-retrieval bounded. However, as in-depth monitoring of interactions during connection attempts becomes important or as connection attacks generate high data rates of traffic moving beyond the TOE to the processor, the gains drop rapidly. In the case of SQL Slammer, servers forwarded User Datagram Protocol (UDP) packets to the central processor, which had to perform payload analysis before forwarding them in turn. Thus, the servers had to deal with interrupt latency, memory bus utilization, and management issues, as well as raw clock cycles as data rates scaled.

To address these limitations, appliances often target a task-specific set of functionalities. The intrusion protection system marketplace is a good example. Designers have used a variety of methodologies, architectures, and feature sets to pioneer new mechanisms that provide stronger security than ACLs and stateful firewalls. As the threats have changed, devices and appliances have introduced new functionality to address them. Although appliances typically can perform a given function effectively, a network composed of many single-function devices increases the complexity. To reduce capital and operational expense, converged devices are beginning to appear, especially in the high-speed network security arena.

Although routers and switches offer performance and scalability, servers dominate the market on flexibility. Appliances have pinpointed "pain-points," such as network-based antivirus, antispam, and URL filtering, to achieve targeted resolution. Without clever software, devices might not be able to adapt to the ever-changing threats. Over time, new breeds of devices will likely enter the market. Neither routers nor servers, these devices will scale and remain flexible while providing network security. Accomplishing this requires leveraging new technologies beyond traditional ASIC- and CPU-centric architectures.

BEYOND TRADITIONAL DESIGNS

As security issues break the OSI model bounds of traditional networking solutions, the systems to address them must do so as well. ASICs and CPUs have traditionally handled networking issues, sometimes combined with exception handlers or offload engines. As automated attacks increase and attack tools grow more sophisticated, however, discovering vulnerabilities and combating asymmetric threats requires a better arsenal of technologies.

In recent years, technologies such as network processors[6] have driven a new breed of solutions that perform at high data rates while remaining flexible through software. Although no single chip can provide the data rate, processing performance, and flexibility to meet every demand, these devices offer many benefits over traditional CPU or ASIC models. By performing stackless operations against packets, network processors improve performance and offer increased flexibility for selecting packets for inspection, thus dramatically reducing system bandwidth use while addressing high data rate problems. However, as many leading network processors move into their third generation, their limitations are becoming well known. Although flexible in their treatment of header fields, they

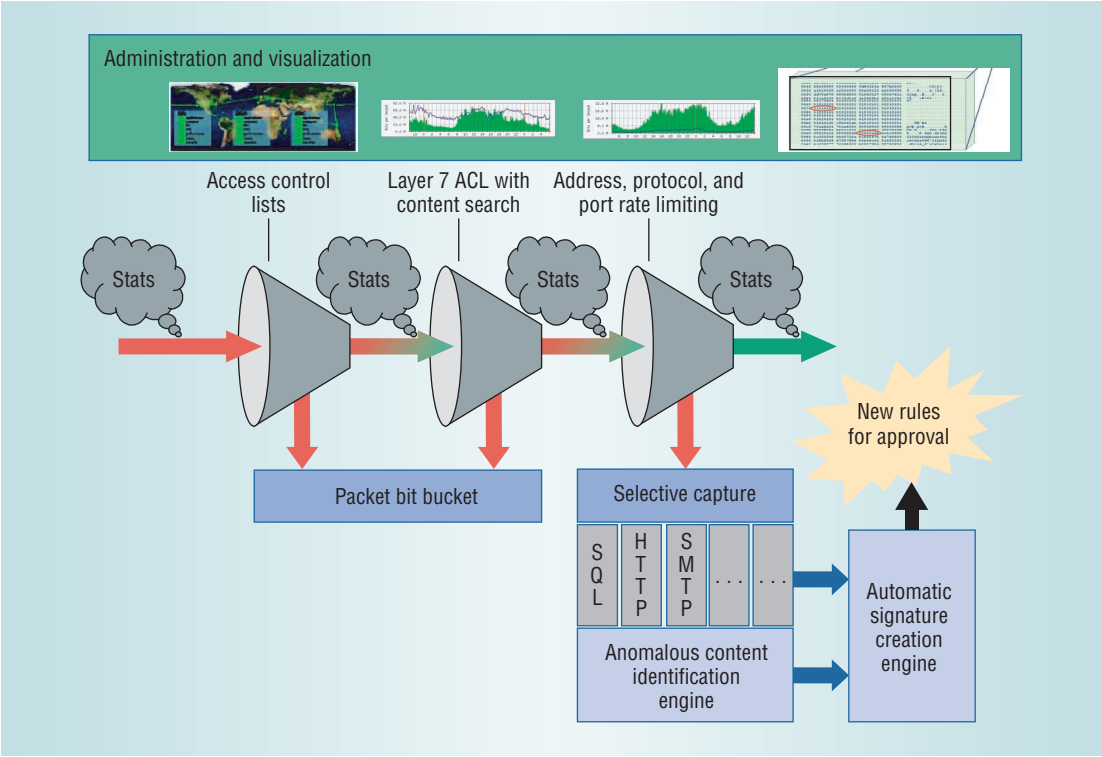


Figure 5. Sample adaptive model. Packet-based security mechanisms send packets to an anomalous content-identification engine, which creates new signatures or access-control enforcement policies.

aren't fast enough to maintain performance on complex network security processing tasks.

Fortunately, with significant advances in configurable logic such as field-programmable gate arrays (FPGAs) and associated custom memories, system designs can provide flexibility while maintaining high performance. Custom coprocessors for performing regular expression processing of payloads and large-table lookups bring the impossible within reason. These processors can change the approach, and ultimately the paradigm, for solving high-speed network security issues.

Network processors and FPGAs provide powerful, high data rate programmable and configurable logic. Ternary content-addressable memories, also called network search engines, provide high-speed table lookups for switching ACLs to user name lists. Classification coprocessors[7] provide bulk payload or field-level analysis and comparison of patterns or keywords to use in conjunction with other correlative algorithms to determine traffic intent. Table 1 summarizes these technologies.

Figure 4 shows how a modern architecture might address a problem like SQL Slammer by identifying a malicious packet without impeding valid packets to the same service. In this case, the network processor retrieves the header fields necessary to identify the packet type. The processor sends key header fields, such as IP address and UDP port number, to a device such as a network search engine to check for matching packets, which may require further analysis. To determine whether a UDP packet to port 1434, for example, contains the SQL Slammer, the classification coprocessor searches its payload for particular patterns or signatures. If the coprocessor search results identify the packet as problematic, the network processor can redirect the packet to the bit bucket or take an alternative course of action.

SOFTWARE AND COLLABORATION

Evolutionary advancement and sheer willpower will lead to gains in high-speed network performance and flexibility over time. However, the cunning of hackers with evil intent will continue to advance at an amazing pace, zigzagging around each advancement. Software solutions must become autonomous, adapting to counteract the threat's speed.

To accomplish this goal, systems must be self-adapting. They must be deployed in topologies that enable response, and they must have software that not only performs access control, preprogrammed signature filtering, and anomaly detection, but also identifies new threats through behavior monitoring and other means. Such self-adapting systems could meet the expected needs of security administrators in a world in which the threat is orders of magnitude greater than the current potential for response.[7]

Future systems will need to integrate multiple security solutions into converged systems, with each element feeding and interacting with its counterparts. On top of the base pipeline of security management technologies, constantly running heuristic engines will analyze every bit of available data for unforeseen threats.

Figure 5 shows a proposed adaptive model that deploys a few common technologies that empower

packet-based security defenses. These technologies feed adjacent *anomalous content-inspection* engines, which dynamically create new signatures or access-control enforcement policies. The packet-based security mechanisms include:

- ACLs;
- layer-7 ACL with content search; and
- address, protocol, and port rate limits.

In the sample adaptive model, traffic that successfully traverses the pipeline is assumed to be good traffic. Should traffic fall outside defined thresholds at any stage, the packet is discarded as an attack and alerts are sent based on severity; traffic that falls outside defined thresholds can also be captured for further analysis. The model could then organize captured traffic into multiple content channels based on protocols and ports—for example, it could maintain captured SQL traffic separately from captured HTTP traffic.

Several technologies can analyze and manage traffic-management statistics and packet-capture channels:

- anomalous content identification,
- automatic signature creation,
- administrator control responses, and
- traffic and what-if visualization.

As they capture data, engines analyze it for patterns common to the captured packets yet unlike previously known normal traffic for the protocol in question. Using fractal-based complexity theory models, these engines can find the needle in the haystack. After identifying a unique pattern, the engine passes it along to other engines that create automatic signatures by analyzing the headers for the protocol and port in question and the tip-off information from the anomalous content-identification engine. The engine can add the resulting signature to the pipeline automatically or after appropriate administrator approval.

New breeds of systems based on innovative processing components will help achieve flexible line-rate high-speed security over time. The problem is beyond silicon, systems, and applications, however. It's in the standards we write without security in mind, the complexities we introduce by not adhering to standards, and the network topologies we continue to paste together without a holistic view. Breaking the legacy approach mold, introducing overlay networks of devices focused on ensuring the infrastructure's security while not impeding its services, and continuing research in self-healing, self-adapting policy enforcement are critical to creating a high-speed network robust enough to defend against equally high-speed attacks. ■

References

1. W.R. Cheswick, S.M. Bellovin, and A.D. Rubin, *Firewalls and Internet Security: Repelling the Wily Hacker*, Addison-Wesley, 2003.
2. A. Freedman, "Securing the Edge," *ACM Queue*, vol. 1, no. 1, 2003; www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=3.
3. Netscreen, "Next-Generation Security Solutions for the Broadband Internet," white paper, Feb. 2000; www.firewallsdirect.com/white_papers/html_White/netscreen_html/next_generation.htm.
4. K. Tolly, "High-Speed Security," The Tolly Group, 2002; www.nortelnetworks.com/solutions/collateral/highspeed_security.pdf.
5. D. Moore et al., *The Spread of the Sapphire/Slammer Worm*, tech. report, Caida, 2003; www.caida.org/outreach/papers/2003/sapphire/sapphire.html.
6. B. Wheeler and L. Gwennap, *A Guide to Network Processors*, 5th ed., The Linley Group, 2003; www.linleygroup.com/Reports/npu_guide.html.
7. J. Bolaria and B. Wheeler, *A Guide to Search Engines and Traffic Managers*, 4th ed., The Linley Group, 2004; www.linleygroup.com/Reports/memory_guide.html.

Peder Jungck *is the chairman, founder, and chief technology officer of CloudShield Technologies. His research interests include network security, high-speed content-based networking system architectures, autonomic networking, and compiler and language design. Jungck received a BA in mathematics and computer science from Beloit College, Wisconsin. Contact him at peder@cloudshield.com.*

Simon S.Y. Shim *is an associate professor in the Department of Computer Engineering at San Jose State University. His research interests include network security, e-commerce, distributed systems, and multimedia databases. Shim received a PhD in computer science from the University of Minnesota. Contact him at sishim@email.sjsu.edu.*

Scaling to the End of Silicon with EDGE Architectures

The TRIPS architecture is the first instantiation of an EDGE instruction set, a new, post-RISC class of instruction set architectures intended to match semiconductor technology evolution over the next decade, scaling to new levels of power efficiency and high performance.

Doug Burger
Stephen W. Keckler
Kathryn S. McKinley
Mike Dahlin
Lizy K. John
Calvin Lin
Charles R. Moore
James Burrill
Robert G. McDonald
William Yoder, and
the TRIPS Team
The University of Texas at Austin

Instruction set architectures have long lifetimes because introducing a new ISA is tremendously disruptive to all aspects of a computer system. However, slowly evolving ISAs eventually become a poor match to the rapidly changing underlying fabrication technology. When that gap eventually grows too large, the benefits gained by renormalizing the architecture to match the underlying technology make the pain of switching ISAs well worthwhile.

Microprocessor designs are on the verge of a post-RISC era in which companies must introduce new ISAs to address the challenges that modern CMOS technologies pose while also exploiting the massive levels of integration now possible. To meet these challenges, we have developed a new class of ISAs, called Explicit Data Graph Execution (EDGE), that will match the characteristics of semiconductor technology over the next decade.

TIME FOR A NEW ARCHITECTURAL MODEL?

The "Architecture Comparisons" sidebar provides a detailed view of how previous architectures evolved to match the driving technology at the time they were defined.

In the 1970s, memory was expensive, so CISC architectures minimized state with dense instruction encoding, variable-length instructions, and small numbers of registers. In the 1980s, the number of devices that could fit on a single chip replaced absolute transistor count as the key limiting resource. With a reduced number of instructions and modes as well as simplified control logic, an entire RISC processor could fit on a single chip. By moving to a register-register architecture, RISC ISAs supported aggressive pipelining and, with careful compiler scheduling, RISC processors attained high performance despite their reduction in complexity.

Since RISC architectures, including the x86 equivalent with µops, are explicitly designed to support pipelining, the unprecedented acceleration of clock rates—40 percent per year for well over a decade—has permitted performance scaling for the past 20 years with only small ISA changes. For example, Intel has scaled its x86 line from 33 MHz in 1990 to 3.4 GHz today—more than a 100-fold increase over 14 years. Approximately half of that increase has come from designing deeper pipelines. However, recent studies show that pipeline scaling is nearly exhausted,[1] indicating that processor design now requires innovations beyond pipeline-centric ISAs. Intel's recent cancellation of its high-frequency Pentium 4 successors is further evidence of this imminent shift.

Future architectures must support four major emerging technology characteristics:

- Pipeline depth limits mean that architects must rely on other fine-grained concurrency mechanisms to improve performance.
- The extreme acceleration of clock speeds has hastened power limits; in each market, future architectures will be constrained to obtain as much performance as possible given a hard

Published by the IEEE Computer Society
0018-9162/04/$20.00 © 2004 IEEE

Architecture Comparisons

Comparing EDGE with historic architectures illustrates that architectures are never designed in a vacuum—they borrow frequently from previous architectures and can have many similar attributes.

- **VLIW:** A TRIPS block resembles a 3D VLIW instruction, with instructions filling fixed slots in a rigid structure. However, the execution semantics differ markedly. A VLIW instruction is statically scheduled—the compiler guarantees when it will execute in relation to all other instructions.[1] All instructions in a VLIW packet must be independent. The TRIPS processor is a static placement, dynamic issue (SPDI) architecture, whereas a VLIW machine is a static placement, static issue (SPSI) architecture. The static issue model makes VLIW architectures a poor match for highly communication-dominated future technologies. Intel's family of EPIC architectures is a VLIW variant with similar limitations.
- **Superscalar:** An out-of-order RISC or x86 superscalar processor and an EDGE processor traverse similar dataflow graphs. However, the superscalar graph traversal involves following renamed pointers in a centralized issue window, which the hardware constructs—adding instructions individually—at great cost to power and scalability. Despite attempts at partitioned variants,[2] the scheduling scope of superscalar hardware is too constrained to place instructions dynamically and effectively. In our scheduling taxonomy, a superscalar processor is a dynamic placement, dynamic issue (DPDI) machine.
- **CMP:** Many researchers have remarked that a TRIPS-like architecture resembles a chip multiprocessor (CMP) with 16 lightweight processors. In an EDGE architecture, the global control maps irregular blocks of code to all 16 ALUs at once, and the instructions execute at will based on dataflow order. In a 16-tile CMP, a separate program counter exists at each tile, and tiles can communicate only through memory. EDGE architectures are finer-grained than both CMPs and proposed speculatively threaded processors.[3]
- **RAW processors:** At first glance, the TRIPS microarchitecture bears similarities to the RAW architecture. A RAW processor is effectively a 2D, tiled static machine, with shades of a highly partitioned VLIW architecture. The main difference between a RAW processor and the TRIPS architecture is that RAW uses compiler-determined issue order (including the inter-ALU routers), whereas the TRIPS architecture uses dynamic issue, to tolerate variable latencies. However, since the RAW ISA supports direct communication of a producer instruction in one tile to a consuming instruction's ALU on another tile, it could be viewed as a statically scheduled variant of an EDGE architecture, whereas TRIPS is a dynamically scheduled EDGE ISA.
- **Dataflow machines:** Classic dataflow machines, researched heavily at MIT in the 1970s and 1980s,[4] bear considerable resemblances to intrablock execution in EDGE architectures. Dataflow machines were originally targeted at functional programming languages, a suitable match because they have ample concurrency and have side-effect free, "write-once" memory semantics. However, in the context of a contemporary imperative language, a dataflow architecture would need to store and process many unnecessary instructions because of complex control-flow conditions. EDGE architectures use control flow between blocks and support conventional memory semantics within and across blocks, permitting them to run traditional imperative languages such as C or Java, while gaining many of the benefits of more traditional dataflow architectures.
- **Systolic processors:** Systolic arrays are a special historical class of multidimensional array processors[5] that continually process the inputs fed to them. In contrast, EDGE processors issue the set of instructions dynamically in mapped blocks, which makes them considerably more general purpose.

References

1. J.A. Fisher et al., "Parallel Processing: A Smart Compiler and a Dumb Machine," *Proc. 1984 SIGPLAN Symp. Compiler Construction*, ACM Press, 1984, pp. 37-47.
2. K.I. Farkas et al., "The Multicluster Architecture: Reducing Cycle Time Through Partitioning," *Proc. 30th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO-30), IEEE CS Press, 1997, pp. 149-159.
3. G.S. Sohi, S.E. Breach, and T.N. Vijaykumar, "Multiscalar Processors," *Proc. 22nd Int'l Symp. Computer Architecture* (ISCA 95), IEEE CS Press, 1995, pp. 414-425.
4. Arvind, "Data Flow Languages and Architecture," *Proc. 8th Int'l Symp. Computer Architecture* (ISCA 81), IEEE CS Press, 1981, p. 1.
5. H.T. Kung, "Why Systolic Architectures?" *Computer*, Jan. 1982, pp. 37-46.

An EDGE ISA provides a richer interface between the compiler and the microarchitecture.

power ceiling; thus, they must support *power-efficient performance*.
- Increasing resistive delays through global on-chip wires means that future ISAs must be amenable to on-chip *communication-dominated execution*.
- Design and mask costs will make running many application types across a single design desirable; future ISAs should support *polymorphism*—the ability to use their execution and memory units in different ways and modes to run diverse applications.

The ISA in an EDGE architecture supports one main characteristic: *direct instruction communication*. Direct instruction communication means that the hardware delivers a producer instruction's output directly as an input to a consumer instruction, rather than writing it back to a shared namespace, such as a register file. Using this direct communication from producers to consumers, instructions execute in dataflow order, with each instruction firing when its inputs are available. In an EDGE architecture, a producer with multiple consumers would specify each of those consumers explicitly, rather than writing to a single register that multiple consumers read, as in a RISC architecture.

The advantages of EDGE architectures include higher exposed concurrency and more power-efficient execution. An EDGE ISA provides a richer interface between the compiler and the microarchitecture: The ISA directly expresses the dataflow graph that the compiler generates internally, instead of requiring the hardware to rediscover data dependences dynamically at runtime, an inefficient approach that out-of-order RISC and CISC architectures currently take.

Today's out-of-order issue RISC and CISC designs require many inefficient and power-hungry structures, such as per-instruction register renaming, associative issue window searches, complex dynamic schedulers, high-bandwidth branch predictors, large multiported register files, and complex bypass networks. Because an EDGE architecture conveys the compile-time dependence graph through the ISA, the hardware does not need to rebuild that graph at runtime, eliminating the need for most of those power-hungry structures. In addition, direct instruction communication eliminates the majority of a conventional processor's register writes, replacing them with more energy-efficient delivery directly from producing to consuming instructions.

TRIPS: AN EDGE ARCHITECTURE

Just as MIPS was an early example of a RISC ISA, the TRIPS architecture is an instantiation of an EDGE ISA. While other implementations of EDGE ISAs are certainly possible, the TRIPS architecture couples compiler-driven placement of instructions with hardware-determined issue order to obtain high performance with good power efficiency. At the University of Texas at Austin, we are building both a prototype processor and compiler implementing the TRIPS architecture, to address the above four technology-driven challenges as detailed below:

- To *increase concurrency*, the TRIPS ISA includes an array of concurrently executing arithmetic logic units (ALUs) that provide both scalable issue width and scalable instruction window size; for example, increasing the processor's out-of-order issue width from 16 to 32 is trivial.
- To attain *power-efficient high performance*, the architecture amortizes the overheads of sequential, von Neumann semantics over large, 100-plus instruction blocks.
- Since future architectures must be heavily partitioned, the TRIPS architecture uses compile-time instruction placement to *mitigate communication delays*, which minimizes the physical distance that operands for dependent instruction chains must travel across the chip and thus minimizes execution delay.
- Because its underlying dataflow execution model does not presuppose a given application computation pattern, TRIPS offers increased *flexibility*. The architecture includes configurable memory banks, which provide a general-purpose, highly programmable spatial computing substrate. The underlying dataflow-like execution model, in which instructions fire when their operands arrive, is fundamental to computation, efficiently supporting vectors, threads, dependence chains, or other computation patterns as long as the compiler can spatially map the pattern to the underlying execution substrate.

To support conventional languages such as C, C++, or Fortran, the TRIPS architecture uses *block-atomic execution*. In this model, the compiler groups instructions into blocks of instructions, each of which is fetched, executed, and committed atomically, similar to the conventional notion of transactions: A block may either be com-

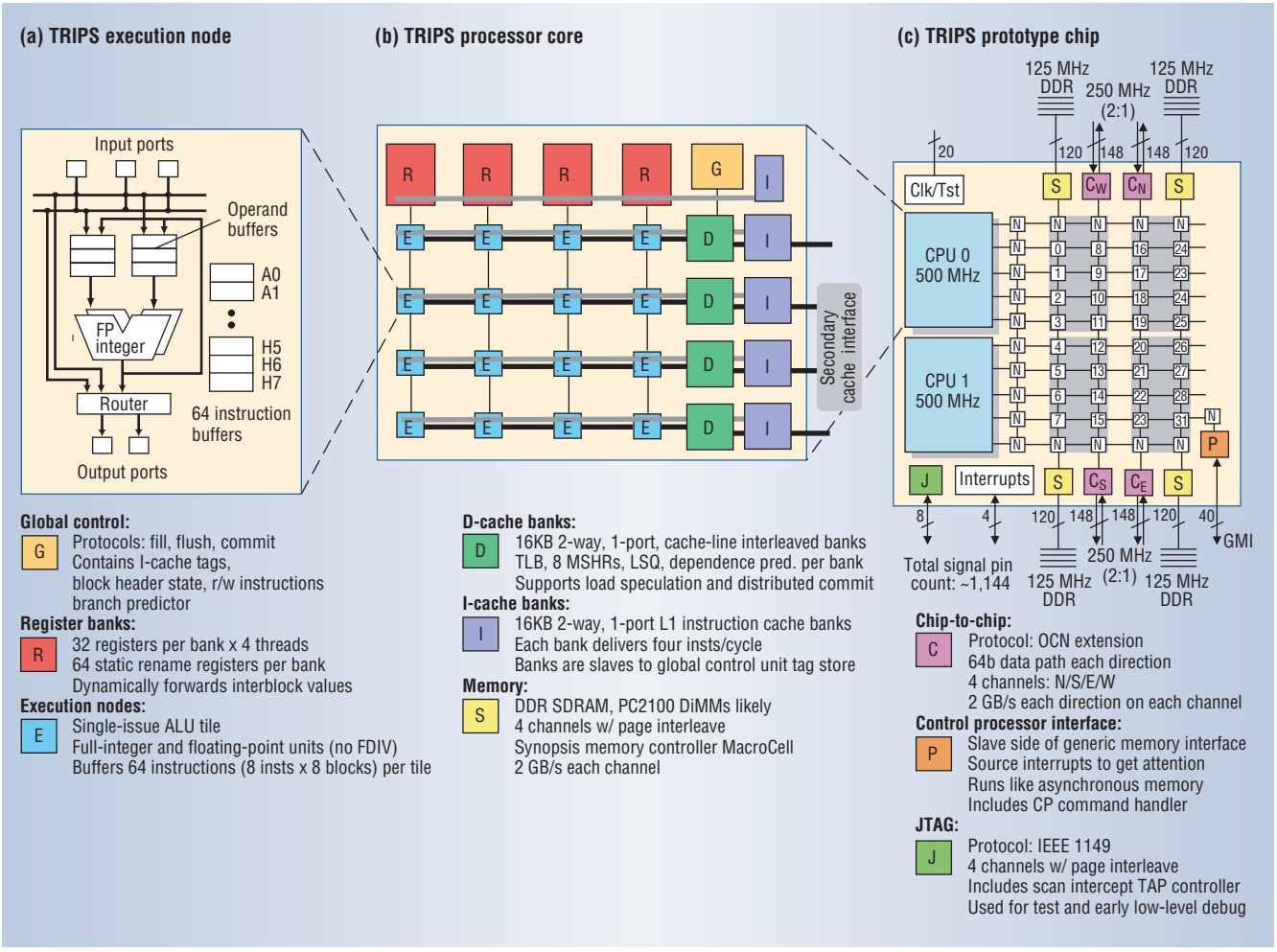


Figure 1. TRIPS prototype microarchitecture. (a) The prototype chip contains two processing cores, each of which is a 16-wide out-of-order issue processor that can support up to 1,024 instructions in flight. Each chip also contains 2 Mbytes of integrated L2 cache, organized as 32 banks connected with a lightweight routing network, as well as external interfaces. (b) The processor core is composed of 16 execution nodes connected by a lightweight network. The compiler builds 128-instruction blocks that are organized into groups of eight instructions per execution node. (c) Each execution node contains a fully functional ALU, 64 instruction buffers, and a router connecting to the lightweight inter-ALU network.

mitted entirely or rolled back; a fraction of a block may not be committed. Each of these TRIPS blocks is a hyperblock[2] that contains up to 128 instructions, which the compiler maps to an array of execution units. The TRIPS microarchitecture behaves like a conventional processor with sequential semantics at the block level, with each block behaving as a "megainstruction." Inside executing blocks, however, the hardware uses a fine-grained dataflow model with direct instruction communication to execute the instructions quickly and efficiently.

TRIPS instructions do not encode their source operands, as in a RISC or CISC architecture. Instead, they produce values and specify where the architecture must route them in the ALU array. For example, a RISC ADD instruction adds the values in R2 and R3, and places the result in R1:

```
ADD R1, R2, R3
```

The equivalent TRIPS instruction specifies only the targets of the add, not the source operands:

```
ADD T1, T2
```

where T1 and T2 are the physical locations (assigned by the compiler) of two instructions dependent on the result of the add, which would have read the result in R1 in the RISC example above. Each instruction thus sends its values to the consumers, and instructions fire as soon as all of their operands arrive.

The compiler statically determines the locations of all instructions, setting the consumer target location fields in each instruction appropriately. Dynamically, the processor microarchitecture fires each instruction as soon as it is ready. When fetching and mapping a block, the processor fetches the instructions in parallel and loads them into the instruction buffers at each ALU in the array. This

TRIPS can achieve power-efficient out-of-order execution across an extremely large instruction window.

block mapping and execution model eliminates the need to go through any fully shared structures, including the register file, for any instruction unless instructions are communicating across distinct blocks. The only exceptions are loads and stores, which must access a bank of the data cache and memory ordering hardware.

To demonstrate the potential of EDGE instruction sets, we are building a full prototype of the TRIPS architecture in silicon, a compiler that produces TRIPS binaries, and a limited runtime system.

Figure 1a is a diagram of the prototype chip, which will be manufactured in 2005 in a 130-nm ASIC process and is expected to run at 500 MHz. The chip contains 2 Mbytes of integrated L2 cache, organized as 32 banks connected with a lightweight routing network. Each of the chip's two processing cores is a 16-wide out-of-order issue processor that can support up to 1,024 instructions in flight, making TRIPS the first kiloinstruction processor to be specified or built.

As Figure 1b shows, each processing core is composed of a 4×4 array of execution nodes connected by a lightweight network. The nodes are not processors; they are ALUs associated with buffers for holding instructions. There are four register file banks along the top, and four instruction and data cache banks along the right-hand side of the core, as well as four ports into the L2 cache network. The compiler builds 128-instruction blocks, organized into groups of eight instructions per node at each of the 16 execution nodes.

To fetch a block of instructions, the global control tile ("G" in Figure 1b) accesses its branch predictor, obtains the predicted block address, and accesses the I-cache tags in the G-tile. If the block's address hits in the I-cache, the G-tile broadcasts the block address to the I-cache banks.

Each bank then streams the block instructions for its respective row into the execution array and into the instruction buffers at each node, shown in Figure 1c. Since branch predictions need occur only once every eight cycles, the TRIPS architecture is effectively unconstrained by predictor or I-cache bandwidth limitations.

Each block specifies register and memory inputs and outputs. Register *read* instructions inject the block inputs into the appropriate nodes in the execution array. Instructions in the block then execute in dataflow order; when the block completes, the control logic writes all register outputs and stores to the register tiles and memory, then it removes the block from the array. Each block emits exactly one branch that produces the location of the subsequent block.

To expose more instruction-level parallelism, the TRIPS microarchitecture supports up to eight blocks executing concurrently. When the G-tile maps a block onto the array, it also predicts the next block, fetching and mapping it as well. In steady state, up to eight blocks can operate concurrently on the TRIPS processor. The renaming logic at the register file bank forwards register values that one block produces directly to consumers in another block to improve performance further.

With this execution model, TRIPS can achieve power-efficient out-of-order execution across an extremely large instruction window because it eliminates many of the power-hungry structures found in traditional RISC implementations. The architecture replaces the associative issue window with architecturally visible instruction buffers constructed from small RAMs. The partitioned register file requires fewer ports because the processor never writes temporary values produced and consumed within a block to the register file. As a result, the TRIPS processor reduces register file accesses and rename table lookups by 70 percent, on average.

The microarchitecture replaces the unscalable broadcast bypass network in superscalar processors with a point-to-point routing network. Although routing networks can be slower than broadcasting, the average distance that operands must travel along the network is short because the compiler places dependent instructions on the same node or on nearby nodes. The instruction cache can provide 16 instructions per cycle and only needs to predict a branch once every eight cycles, reducing the necessary predictor bandwidth. The TRIPS ISA thus amortizes the overheads of out-of-order execution over a 128-instruction block (and with eight blocks, over a 1,024-instruction window), rather than incurring them on every single instruction.

BLOCK COMPILATION

Figure 2 shows an example of how the TRIPS ISA encodes a small, simple compiled block. We assume that the C code snippet in Figure 2a allocates the input integer y to a register. Figure 2b is the same example in MIPS-like assembly code. The variable x is saved in register 2.

The TRIPS compiler constructs the dataflow graph shown in Figure 2c. Two read instructions obtain the register values and forward them to their

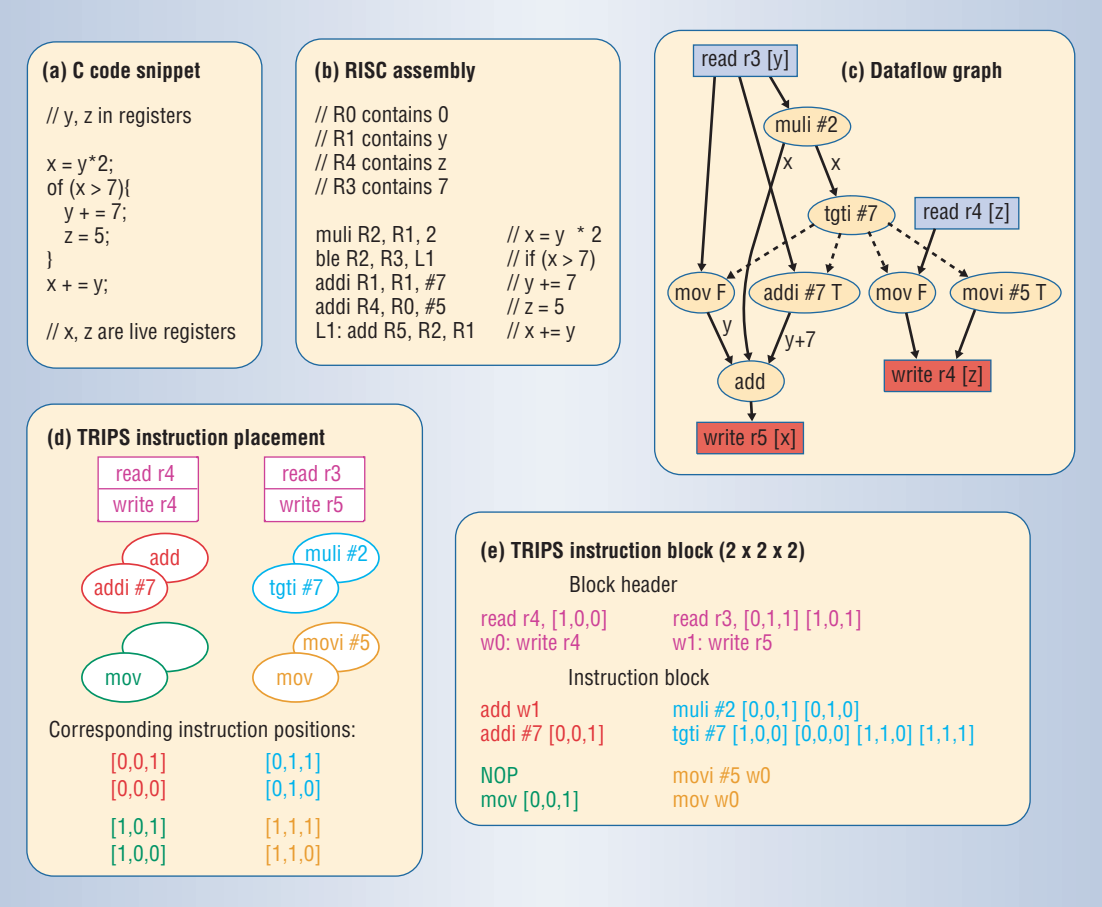


Figure 2. TRIPS code example. (a) In the C code snippet, the compiler allocates the input integer y to a register. (b) In the MIPS-like assembly code, the variable x is saved in register 5. (c) The compiler converts the original branch to a test instruction and uses the result to predicate the control-dependent instructions, which appear as dotted lines in the dataflow graph. (d) Each node in the 2 × 2 execution node array holds up to two buffered instructions. (e) The compiler generates in target form, which correspond to the map of instruction locations at the bottom of part (d).

consuming instructions mapped on the execution array. The compiler converts the original branch to a test instruction and uses the result to predicate the control-dependent instructions, shown as dotted lines. By converting branches to predicates, the compiler creates larger regions with no control flow changes for more effective spatial scheduling.

At runtime, the instructions fire in any order subject to dataflow constraints. The compiled code eventually forwards the block's outputs to the two write instructions and to any other block waiting for those values. Figure 2d shows the dataflow graph (DFG) with the block of instructions scheduled onto a 2 × 2 execution node array, each with an ALU, with up to two instructions buffered per node.

Since the critical path is muli→tgti→addi→add, the compiler assigns the first two instructions to the same ALU so that they can execute with no internode routing delays. No implicit ordering governs the execution of instructions other than the dataflow arcs shown in Figure 2c, so there are no "program order" limitations to instruction issue. When both writes arrive at the register file, the control logic deallocates the block and replaces it with another.

Figure 2e shows the actual code that a TRIPS compiler would generate, which readers can decode using the map of instruction locations at the bottom of Figure 2d. Instructions do not contain their source operands—they contain only the physical locations of their dependent consumers. For example, when the control logic maps the block, the read instruction pulls the value out of register R4 and forwards it to the instruction located in the slot 0 of ALU node [1,1]. As soon as that instruction receives the result of the test condition and the register value, it forwards the value back to register write instruction 0, which then places it back into the register file (R4). If the predicate has a value of false, the instruction doesn't fire.

While this block requires a few extra overhead instructions compared to a RISC ISA, it performs fewer register file accesses—two reads and two writes, as opposed to six reads and five writes. Larger blocks typically have fewer extra overhead instructions, more instruction-level parallelism, and a larger ratio of register file access savings.

COMPILING FOR TRIPS

Architectures work best when the subdivision of labor between the compiler and the microarchitecture matches the strengths and capabilities of each. For future technologies, current execution models strike the wrong balance: RISC relies too little on the compiler, while VLIW relies on it too much.

RISC ISAs require the hardware to discover instruction-level parallelism and data dependences dynamically. While the compiler could convey the

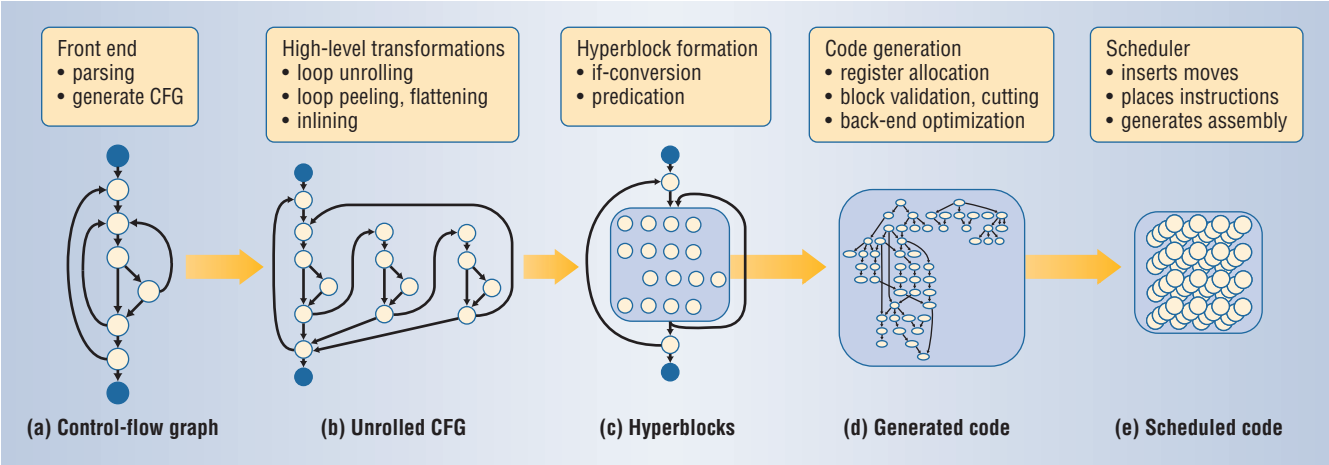


Figure 3. EDGE optimizations in the Scale compiler. (a) A front end parses the code and constructs a control-flow graph (CFG). (b) The CFG after the compiler has unrolled it three times. (c) All of the basic blocks from the unrolled if-converted loop inside a single hyperblock. (d) After inserting the predicates, the compiler generates the code and allocates registers. (e) The scheduler attempts to place independent instructions on different ALUs to increase concurrency and dependent instructions near one another to minimize routing distances and communication delays.

dependences, the ISA cannot express them, forcing out-of-order superscalar architectures to waste energy reconstructing that information at runtime.

VLIW architectures, conversely, put too much of a load on the compiler. They require the compiler to resolve all latencies at compile time to fill instruction issue slots with independent instructions. Since unanticipated runtime latencies cause the machine to stall, the compiler's ability to find independent instructions within its scheduling window determines overall performance. Since branch directions, memory aliasing, and cache misses are unknown at compile time, the compiler cannot generate schedules that best exploit the available parallelism in the face of variable latency instructions such as loads.

For current and future technologies, EDGE architectures and their ISAs provide a proper division between the compiler and architecture, matching their responsibilities to their intrinsic capabilities, and making the job of each simpler and more efficient. Rather than packing together independent instructions like a VLIW machine, which is difficult to scale to wider issue, an EDGE compiler simply expresses the data dependences through the ISA. The hardware's execution model handles dynamic events like variable memory latencies, conditional branches, and the issue order of instructions, *without* needing to reconstruct any compile time information.

An EDGE compiler has two new responsibilities in addition to those of a classic optimizing RISC compiler. The first is forming large blocks (hyperblocks in the TRIPS architecture) that have no internal control flow, thus permitting them to be scheduled as a unit. The second is the spatial scheduling of those blocks, in which the compiler statically assigns instructions in a block to ALUs in the execution array, with the goal of reducing interinstruction communication distances and exposing parallelism.

To demonstrate the tractability of the compiler analyses needed for EDGE architectures, we retargeted the Scale research compiler[3] to generate optimized TRIPS code. Scale is a compilation framework written in Java that was originally designed for extensibility and high performance with RISC architectures such as Alpha and Sparc.

Scale provides classic scalar optimizations and analysis such as constant propagation, loop invariant code motion, dependence analysis, and higher-level transformations such as inlining, loop unrolling, and interchange. To generate high-quality TRIPS binaries, we added transformations to generate large predicated hyperblocks,[2] a new back end to generate *unscheduled* TRIPS code, and a scheduler that maps instructions to ALUs and generates *scheduled* TRIPS assembly in which every instruction is assigned a location on the execution array.

Figure 3 shows the EDGE-specific transformations on the code fragment in Figure 4. Scale uses a front end to parse the code and construct the abstract syntax tree and control-flow graph (CFG), shown in Figure 3a. To form large initial regions, the compiler unrolls loops and inlines functions.

Figure 3b shows the CFG after the compiler has unrolled the inner do-while loop three times. The compiler must take four additional TRIPS-specific steps to generate correct TRIPS binaries.

Large hyperblock formation. The compiler exposes parallelism by creating large predicated hyperblocks, which it can fetch en masse to fill the issue window quickly. Hyperblocks are formally defined as single-entry, multiple-exit regions of code. Because they have internal control-flow transfers, hyperblocks are ideal for mapping onto a spatial substrate. Branches can only jump out of a hyperblock to the start of another hyperblock, never to

```
for (i = 0; i< loopcount; i++) {
  code = 0x1234;
  len = (i % 15) + 1;
  res = 0;
  do {
    res |= code & 1;
    if (res & 0xfddd) res <<= 1;
      code >>= 1,
    } while (--len > 0);
  result += res;
}
```

Figure 4. Modified code fragment from gzip (SPECINT2000).

another point in the same hyperblock or into the middle of another. Large hyperblocks are desirable because they provide a larger region for scheduling and expose more concurrency.

For simplicity, the TRIPS prototype ISA supports fixed 128-instruction hyperblocks, which it fetches at runtime to provide eight instructions at each of the 16 execution nodes. A more complex implementation might permit variable-sized blocks, which map varying numbers of instructions to execution nodes.

To grow hyperblocks, Scale uses inlining, aggressive unrolling (including unrolled while loops), loop peeling/flattening, and if-conversion (a form of predication). Figure 3c shows all the basic blocks from the unrolled and if-converted loop, placed inside a single hyperblock.

At runtime, many hyperblocks will contain useless instructions from CFG nodes included in the hyperblock but not on the dynamically taken path, or from loop iterations unrolled past the end of a loop. To reduce the fraction of unneeded instructions in hyperblocks, Scale uses edge profiling to guide the high-level transformations by identifying both loop counts and infrequently executed basic blocks.

Predicated execution. In previous predication models, evaluating the predicate on every conditional instruction incurs no additional cost. The TRIPS architecture treats a predicate as a dataflow operand that it explicitly routes to waiting predicated consumers. A naive implementation would route a predicate to every instruction in a predicated basic block, creating a wide fan-out problem.

Fortunately, given a dependence chain of instructions that execute on the same predicate, the compiler can choose to predicate only the first instruction in the chain, so the chain does not fire, or only the last instructions in the chain that write registers or memory values. Predicating the first instruction saves power when the predicate is false. Alternatively, predicating only the last, output-producing instructions in a dependence chain generally increases both power and performance by hiding the latency of the predicate computation.

Since the control logic detects block termination when it receives all outputs (stores, register writes, and a branch), in the TRIPS architecture a block must emit the same number of outputs no matter which predicated path is taken. Consequently, the TRIPS ISA supports the notion of *null stores* and *null register writes* that execute whenever a predicated store or register write does not fire. The TRIPS compiler inserts the null assignments optimally using the predicate flow graph, which it generates internally after leaving static single assignment (SSA) form. After inserting the predicates, the compiler generates the code and allocates registers, as shown in Figure 3d, where the hyperblock's dataflow graph is visible.

Generating legal hyperblocks. Scale must confirm that hyperblocks adhere to their legal restrictions before scheduling them onto the execution array. The TRIPS prototype places several restrictions on legal hyperblocks to simplify the hardware: A hyperblock can have no more than 128 instructions, 32 loads or stores along any predicated path, 32 register inputs to the block, and 32 register outputs from the block. These architectural restrictions simplify the hardware at the expense of some hyperblocks being less full to avoid violating those constraints. If the compiler discovers an illegal hyperblock, it splits the block into multiple blocks, allocating intrablock dataflow values and predicates to registers.

Physical placement. After the compiler generates the code, optimizes it, and validates the legality of the hyperblocks, it maps instructions to ALUs and converts instructions to target form, inserting copy operations when a producer must route a value to many consumers. To schedule the code, the compiler assigns instructions to the ALUs on the array, limiting each ALU to at most eight instructions from the block. The scheduler, shown at a high level in Figure 3e, attempts to balance between two competing goals:

- placing independent instructions on different ALUs to increase concurrency, thereby reducing the probability of two instructions competing to issue on the same ALU in the same cycle; and
- placing instructions near one another to minimize routing distances and thus communication delays.

The scheduler additionally exploits its knowledge of the microarchitecture by placing instructions that use the registers near the register banks and by placing critical load instructions near the data cache

A single EDGE architecture can support the three main parallelism classes on the same hardware with the same ISA.

banks. The compiler uses a classic greedy technique to choose the order of instructions to place, with a few additional heuristics to minimize routing distances and ALU contention.

Although the past two years of compiler development have been labor intensive, the fact that we could design and implement this functionality in Scale with a small development team demonstrates the balance in the architecture. The division of responsibilities between the hardware and compiler in the TRIPS architecture is well suited to the compiler's inherent capabilities. Scale can presently compile C and Fortran benchmarks into fully executable TRIPS binaries.

SUPPORTING PARALLELISM

Our work with the TRIPS architecture has shown that a single EDGE architecture can effectively support the three main parallelism classes—instruction, data, and thread—on the same hardware with the same ISA. It may be possible to leverage this inherent flexibility to merge previously distinct markets, such as signal processing and desktop computing, into a single family of architectures that has the same or similar instruction sets and execution models. Area, power, and clock speeds would differentiate implementations to target various power/performance points in distinct markets.

To be fully general and to exploit many types of parallelism, the TRIPS microarchitecture must support the common types of graphs and communication flows across instruction-parallel, data-parallel, and thread-parallel applications without requiring too much specialized hardware support. In previous work, we showed how the TRIPS architecture can harvest instruction-level parallelism from desktop-style codes with complex dependence patterns and, with only minimal additional hardware support, can also support loop-driven data-level parallel codes and threaded parallelism.[4]

Mapping data-level parallelism

Data-level parallel (DLP) applications range from high-end, scientific vector code to graphic processing to low-power, embedded signal processing code. These applications are characterized by frequent, high-iteration loops, large amounts of parallel arithmetic operations, and regular, high-bandwidth access to data sets.

Even though many DLP codes are extremely regular, some such workloads, particularly in the graphics and embedded domains, are becoming less regular, with more complex control flow. EDGE architectures are well suited to both regular and irregular DLP codes because the compiler can map the processing pipelines for multiple data streams to groups of processing elements, using efficient local communication and dataflow-driven activation for execution.

Three additional mechanisms provide significant further benefits to DLP codes. If the compiler can fit a loop body into a single block, or across a small number of blocks, the processor only needs to fetch those instructions once from the I-cache. Subsequent iterations can reuse prior mappings for highly power-efficient loop execution.

In its memory system, the TRIPS prototype has a lightweight network embedded in the cache to support high-bandwidth routing to each of its 32 L2 banks, and it supports dynamic bank configuration using a cache as an explicitly addressable scratchpad. Selectively configuring such memories as scratchpads enables explicit memory management of small on-chip RAMs, similar to signal processing chips such as those in the Texas Instruments C6x series. Our studies show that some DLP codes can benefit directly from higher local memory bandwidth, which developers can add to a TRIPS system by augmenting it with high-bandwidth memory access channels between the processing core and explicitly managed memory. While we will not implement these channels in the prototype, designers may include them in subsequent implementations.

In prior work, we found that adding a small number of additional "universal" hardware mechanisms allowed DLP codes to run effectively on a TRIPS-like processor across many domains such as network processing and cryptography, signal processing, and scientific and graphics workloads.[5] With this additional hardware, a TRIPS-like EDGE processor would, across a set of nine highly varied DLP applications, outperform the best-in-class specialized processor for four applications, come close in two, and fall short in three.

Mapping thread-level parallelism

Most future high-end processors will contain some form of multithreading support. Intel's Pentium 4 and IBM's Power 5 both support simultaneous multithreading (SMT).[6]

We believe that executing a single thread per EDGE processing core will often be more desirable than simultaneously executing multiple threads per core because the EDGE dataflow ISA exposes enough parallelism to effectively use the

core's resources. However, some markets—particularly servers—have copious amounts of thread-level parallelism available, and an EDGE processor should exploit the thread-level parallelism for these applications.

The TRIPS prototype will support simultaneous execution of up to four threads per processing core. The TRIPS multithreading model assigns a subset of the in-flight instruction blocks to each thread. For example, while a single thread might have eight blocks of 128 instructions per block in flight, threads in a four-thread configuration will each have two blocks of 128 instructions in flight.

The processor's control logic moves into and out of multithreading mode by writing to control registers that allocate blocks to threads. In addition to this control functionality, each processor needs a separate copy of an architectural register file for each active thread, as well as some per-thread identifiers augmenting cache tags and other state information.

While the TRIPS prototype maps threads across blocks, thus permitting each thread to have access to all ALUs in the execution array, different EDGE implementations might support other mappings. For example, the hardware could allocate one thread to each column or row of ALUs, thus giving them private access to a register or cache bank. However, these alternative mappings would require more hardware support than the block-based approach used in the TRIPS prototype.

Software challenges

The hardware required to support TLP and DLP applications effectively on an EDGE architecture is but a small increment over mechanisms already present to exploit ILP, making EDGE architectures a good match for exploiting broad classes of parallelism—with high-power efficiency—on a single design. Exploiting an EDGE architecture's flexibility to run heterogeneous workloads comprising a mix of single-threaded, multithreaded, and DLP programs will require modest additional support from libraries and operating systems.

In the TRIPS prototype, memory banks can be configured as caches or as explicitly addressable scratchpads, which lack the process-ID tags that allow processes to share virtually addressed caches. Therefore, to context switch-out/switch-in a process that has activated scratchpad memory, the TRIPS runtime system must save and restore the contents of the scratchpad and reconfigure the memory banks to the appropriate mode of operation. Additionally, to run a heterogeneous mix of jobs on the TRIPS prototype efficiently, the scheduler software should configure processors between single-threaded mode (for most ILP processes and DLP processes) and multithreaded mode (for processes that expose parallelism via large numbers of threads). Further, thread scheduler policies should account for the differing demands of various types of processes.

Overall, because the underlying EDGE substrate provides an abstraction that maps well to a range of different parallelism models, the architecture provides a good balance between hardware and software complexity. Hardware extensions to support ILP, DLP, and TLP are modest, and mechanisms and policies for switching among modes of computation are tractable from a software perspective.

Major ISA changes are traumatic for industry, but solving current challenges may require a major architectural shift.

TO CMP OR NOT TO CMP?

The semiconductor process community has done its job all too well: Computer architects face daunting and interlocking challenges. Reductions in feature size have provided tremendous opportunities by increasing transistor counts, but these advances have introduced new problems of communication delay and power consumption. We believe that finding the solution to these fundamental issues will require a major architecture shift and that EDGE architectures are a good match for meeting these challenges.

Despite the advantages that EDGE architectures offer, major ISA changes are traumatic for industry, especially given the complexity that systems have accrued since the last major shift. However, since then many institutions and companies have developed the technology to incorporate such a new architecture under the hood. For example, Transmeta's code morphing software dynamically translates x86 instructions into VLIW code for its processors. Dynamic translation to an EDGE architecture will likely be simpler than to VLIW, making this technology a promising candidate for solving ISA backward compatibility. We are beginning work on such an effort and have already built a simple PowerPC-to-TRIPS static binary translator.

The competing approach for future systems is explicitly parallel hardware backed by parallelizing software. IBM and Sun Microsystems are both moving to chip multiprocessor (CMP)[7] and chip multithreading models in which each chip contains many processing cores and thread slots that exploit explicitly parallelized threads. Other research efforts, such as Smart Memories[8] and Imagine[9] at

Stanford and RAW[10] at MIT, support DLP workloads with the copious explicit parallelism that software can obtain. This camp argues that future workloads will inevitably shift to be highly parallel and that programmers or compilers will be able to map the parallelism in tomorrow's applications onto a simple, explicitly parallel hardware substrate. Although researchers have consistently made this argument over the past 30 years, the general-purpose market has instead, every time, voted in favor of larger, more powerful uniprocessor cores. The difficulty of scaling out-of-order RISC cores, coupled with IBM's thread-rich server target market, have together driven the emergence of CMPs such as Power 4.

EDGE architectures offer an opportunity to scale the single-processor model further, while still effectively exploiting DLP and TLP when the software can discover it. However, because of the difficulty of discovering and exploiting parallelism, we expect that software will make better use of smaller numbers of more powerful processors. For example, 64 simple processors are much less desirable than four processors, each of which are eight times more powerful than the simple processors. EDGE architectures appear to offer a progressively better solution as technology scales down to the end of silicon, with each generation providing a richer spatial substrate at the expense of increased global communication delays. EDGE ISAs may also be a good match for postsilicon devices, which will likely be communication-dominated as well.

Whether EDGE architectures prove to be a compelling alternative will depend on their performance and power consumption relative to current high-end devices. The prototype TRIPS processor and compiler will help to determine whether this is the case. We expect to have reliable simulation results using optimized compiled code in mid-2004, tape-out in the beginning of 2005, and full chips running in the lab by the end of 2005. ■

Acknowledgments

We thank the following student members of the TRIPS team for their contributions as coauthors of this article: Xia Chen, Rajagopalan Desikan, Saurabh Drolia, Jon Gibson, Madhu Saravana Sibi Govindan, Paul Gratz, Heather Hanson, Changkyu Kim, Sundeep Kumar Kushwaha, Haiming Liu, Ramadass Nagarajan, Nitya Ranganathan, Eric Reeber, Karthikeyan Sankaralingam, Simha Sethumadhavan, Premkishore Sivakumar, and Aaron Smith.

This research is supported by the Defense Advanced Research Projects Agency under contracts F33615-01-C-1892 and F33615-03-C-4106, NSF infrastructure grant EIA-0303609, NSF CAREER grants CCR-9985109 and CCR-9984336, two IBM University Partnership awards, an IBM Shared University Research grant, as well as grants from the Alfred P. Sloan Foundation and the Intel Research Council.

References

1. M.S. Hrishikesh et al., "The Optimal Logic Depth per Pipeline Stage Is 6 to 8 fo4 Inverter Delays," *Proc. 29th Int'l Symp. Computer Architecture* (ISCA 02), IEEE CS Press, 2002, pp. 14-24.
2. S.A. Mahlke et al., "Effective Compiler Support for Predicated Execution Using the Hyperblock," *Proc. 25th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO-25), IEEE CS Press, 1992, pp. 45-54.
3. R.A. Chowdhury et al., "The Limits of Alias Analysis for Scalar Optimizations," *Proc. ACM SIGPLAN 2004 Conf. Compiler Construction*, ACM Press, 2004, pp. 24-38.
4. K. Sankaralingam et al., "Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture," *Proc. 30th Int'l Symp. Computer Architecture* (ISCA 03), IEEE CS Press, 2003, pp. 422-433.
5. K. Sankaralingam et al., "Universal Mechanisms for Data-Parallel Architectures," *Proc. 36th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO-36), IEEE CS Press, 2003, pp. 303-314.
6. D.M. Tullsen, S.J. Eggers, and H.M. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proc. 22nd Int'l Symp. Computer Architecture* (ISCA 95), IEEE CS Press, 1995, pp. 392-403.
7. K. Olukotun et al., "The Case for a Single-Chip Multiprocessor," *Proc. 6th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 94), ACM Press, 1994, pp. 2-11.
8. K. Mai et al., "Smart Memories: A Modular Reconfigurable Architecture," *Proc. 27th Int'l Symp. Computer Architecture* (ISCA 00), IEEE CS Press, 2000, pp. 161-171.
9. S. Rixner et al., "A Bandwidth-Efficient Architecture for Media Processing," *Proc. 31st Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO-31), IEEE CS Press, 1998, pp. 3-13.
10. E. Waingold et al., "Baring It All to Software: RAW Machines," *Computer*, Sept. 1997, pp. 86-93.

Doug Burger, Stephen W. Keckler, Kathryn S. McKinley, *and* ***Mike Dahlin*** *are associate professors in the Department of Computer Sciences at the University of Texas at Austin. They are members of the ACM and senior members of the IEEE. Contact them at {dburger, skeckler, mckinley, dahlin}@cs.utexas.edu.*

Lizy K. John *is an associate professor in the Department of Electrical and Computer Engineering at the University of Texas at Austin. She is a member of the ACM and a senior member of the IEEE. Contact her at ljohn@ece.utexas.edu.*

Calvin Lin *is an associate professor in the Department of Computer Sciences at the University of Texas at Austin and is a member of the ACM. Contact him at lin@cs.utexas.edu.*

Charles R. Moore*, a senior research fellow at the University of Texas at Austin from 2002 through 2004, is currently a Senior Fellow at Advanced Micro Devices. Contact him at chuck.moore@amd.com.*

James Burrill *is a research fellow in the Computer Science Department at the University of Massachusetts at Amherst. Contact him at burrill@cs.umass.edu.*

Robert G. McDonald *is the chief engineer for the TRIPS prototype chip at the University of Texas at Austin. Contact him at robertmc@cs.utexas.edu.*

William Yoder *is a research programmer at the University of Texas at Austin and a member of the ACM and the IEEE. Contact him at byoder@cs.utexas.edu.*

Composing Adaptive Software

Compositional adaptation enables software to modify its structure and behavior dynamically in response to changes in its execution environment. A review of current technology compares how, when, and where recomposition occurs.

Philip K. McKinley
Seyed Masoud Sadjadi
Eric P. Kasten
Betty H.C. Cheng
Michigan State University

Interest in adaptive computing systems has increased dramatically in the past few years, and a variety of techniques now allow software to adapt dynamically to its environment. Two revolutions in the computing field are driving this development. First is the emergence of *ubiquitous computing*,[1] which focuses on dissolving traditional boundaries for how, when, and where humans and computers interact. For example, mobile computing devices must adapt to variable conditions on wireless networks and conserve limited battery life. Second is the growing demand for *autonomic computing*,[2] which focuses on developing systems that can manage and protect themselves with only high-level human guidance. This capability is especially important to systems such as financial networks and power grids that must survive hardware component failures and security attacks.

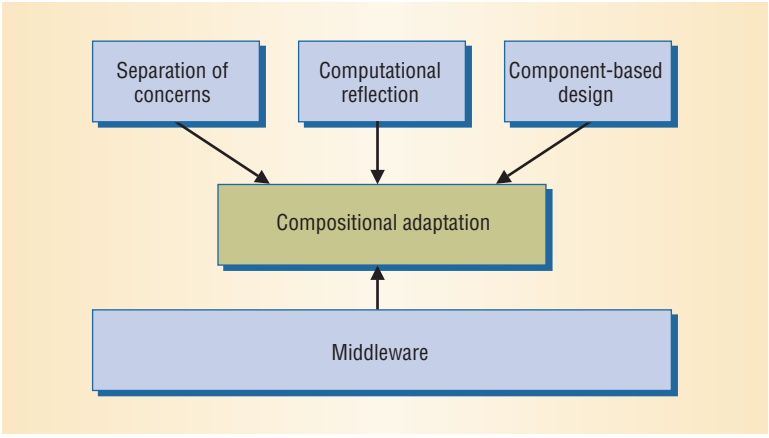


Figure 1. Main technologies supporting compositional adaptation: separation of concerns, computational reflection, and component-based design.

There are two general approaches to implementing software adaptation. *Parameter adaptation* modifies program variables that determine behavior. The Internet's Transmission Control Protocol is an often-cited example: TCP adjusts its behavior by changing values that control window management and retransmissions in response to apparent network congestion. But parameter adaptation has an inherent weakness. It does not allow new algorithms and components to be added to an application after the original design and construction. It can tune parameters or direct an application to use a different existing strategy, but it cannot adopt new strategies.

By contrast, *compositional adaptation* exchanges algorithmic or structural system components with others that improve a program's fit to its current environment. With compositional adaptation, an application can adopt new algorithms for addressing concerns that were unforeseen during development. This flexibility supports more than simple tuning of program variables or strategy selection. It enables dynamic recomposition of the software during execution—for example, to switch program components in and out of a memory-limited device or to add new behavior to deployed systems.

Dynamic recomposition of software dates back to the earliest days of computing, when self-modifying code supported runtime program optimization and explicit management of physical memory. However, such programs were difficult to write and debug. Several new software tools and technologies now help address these problems. Given the increasing pace of research in compositional adaptation, we offer a review of the supporting technologies, proposed solutions, and areas that require further study.

Published by the IEEE Computer Society

Middleware and Adaptation

Much recent research in adaptive software focuses on middleware—the layers of services separating applications from operating systems and network protocols.

Douglas Schmidt decomposes middleware into four layers,[1] shown in Figure A:

- *Host-infrastructure middleware* resides atop the operating system and provides a high-level API that hides the heterogeneity of hardware devices, operating systems, and—to some extent—network protocols.
- *Distribution middleware* provides a high-level programming abstraction, such as remote objects, enabling developers to write distributed applications in a way similar to stand-alone programs. Corba, DCOM, and Java RMI all fit in this layer.
- *Common middleware* services include fault tolerance, security, persistence, and transactions involving entities such as remote objects.
- *Domain-specific middleware* services are tailored to match a particular class of applications.

Most adaptive middleware is based on an object-oriented programming paradigm and derived from popular middleware platforms such as Corba, Java RMI, and DCOM/.NET.

Many adaptive middleware approaches work by intercepting and modifying messages. Figure B shows the flow of a request-reply sequence in a simplified Corba client-server application. This application comprises two autonomous programs hosted on two computers connected by a network.

Assume that the *client* has a valid Corba reference to the *servant* object. The client request to the servant goes first to the *stub*, which represents the Corba object on the client side. The stub marshals the request and sends it to the client *object request broker*. The client ORB sends the request to the server ORB, where a *skeleton* unmarshals the request and delivers it to the servant. The servant replies to the request, by way of the server ORB and skeleton. The client ORB will receive the reply and dispatch it to the client.

In recent years, numerous studies have addressed the issue of how middleware can adapt to dynamic, heterogeneous environments to better serve applications.[2,3] Middleware traditionally hides resource distribution and platform heterogeneity from the application business logic. Thus it is a logical place to put adaptive behavior that is related to crosscutting concerns such as QoS, energy management, fault tolerance, and security policy.

References

1. D.C. Schmidt, "Middleware for Real-Time and Embedded Systems," *Comm. ACM*, June 2002, pp. 43-48.
2. *Comm. ACM*, special issue on adaptive middleware, June 2002, pp. 30-64.
3. *IEEE Distributed Systems Online*, special issue on reflective middleware, June 2001; http://dsonline.computer.org/0105/features/gei0105.htm.

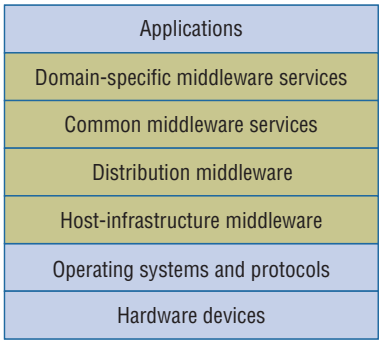


Figure A. Four-layer decomposition of middleware to bridge the gap between an application program and the underlying operating systems, network protocols, and hardware devices.

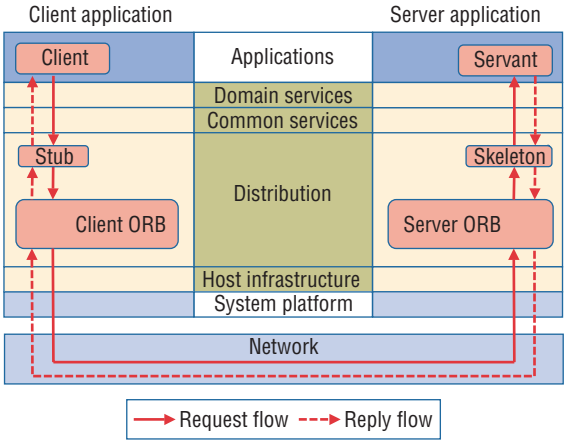


Figure B. Corba call sequence for a simplified client-server application.

ENABLING TECHNOLOGIES

At the core of all approaches to compositional adaptation is a level of indirection for intercepting and redirecting interactions among program entities. Figure 1 shows three technologies—separation of concerns, computational reflection, and component-based design—that we consider as key to reconfigurable software design. Programmers can use these technologies to construct self-adaptive systems in a systematic and principled—as opposed to ad hoc—manner.[3]

In addition, the widespread use of middleware in distributed computing has been a catalyst for compositional adaptation research. Middleware provides a natural place to locate many types of adaptive behavior, as the "Middleware and Adaptation" sidebar describes.

Separation of concerns

Separation of concerns[4] enables the separate development of an application's functional behavior—that is, its business logic—and the code for

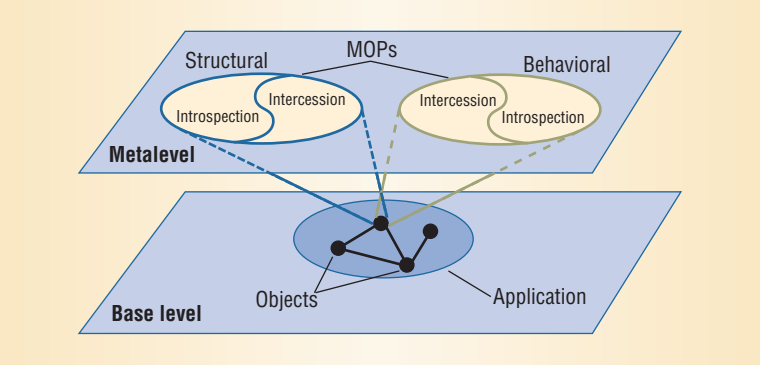


Figure 2. Metalevel understanding collected into metaobject protocols.

crosscutting concerns, such as quality of service (QoS), energy consumption, fault tolerance, and security. An application cannot implement a crosscutting concern at a single program location; instead, it must add the code at many places. Separating crosscutting concerns from functional behavior simplifies development and maintenance, while promoting software reuse.

Separation of concerns has become an important principle in software engineering.[5] Presently, the most widely used approach appears to be aspect-oriented programming.[6] AOP provides abstraction techniques and language constructs to manage crosscutting concerns. The code implementing these concerns, called *aspects*, is developed separately from other parts of the system. In AOP, *pointcuts* are sets of locations in the code where the developer can weave in aspects. Pointcuts are typically identified during development. Later, for example during compilation, the developer uses a specialized compiler, called an *aspect weaver*, to combine different aspects with an application's business logic to create a program with new behavior. An example is the AspectJ compiler. AOP proponents argue that disentangling crosscutting concerns leads to simpler software development, maintenance, and evolution.

AOP is important to dynamic recomposition because most adaptations are relative to some crosscutting concern, such as QoS. AOP enables these concerns to be isolated from the rest of the program. However, in traditional AOP the compiled program is still tangled. To support dynamic recomposition, the programmer needs a way to maintain this separation at runtime.

Computational reflection

Computational reflection refers to a program's ability to reason about, and possibly alter, its own behavior.[7] Reflection enables a system to reveal selected details of its implementation without compromising portability.

Reflection comprises two activities: *introspection* to let an application observe its own behavior, and *intercession* to let a system or application act on these observations and modify its own behavior. In a self-auditing distributed application, for example, software "sensors" could use introspection to observe and report usage patterns for various components. Intercession would allow the system to insert new types of sensors, as well as components that implement corrective action, at runtime.

As Figure 2 shows, a reflective system (represented as base-level objects) and its self-representation (represented as metalevel objects) are causally connected, meaning that modifications to either one will be reflected in the other.

A metaobject protocol (MOP) is an interface that enables "systematic" introspection and intercession of the base-level objects. MOPs support either structural or behavioral reflection.[3] *Structural reflection* addresses issues related to class hierarchy, object interconnection, and data types. As an example, a metalevel object can examine a base-level object to determine what methods are available for invocation. Conversely, *behavioral reflection* focuses on the application's computational semantics. For instance, a distributed application can use behavioral reflection to select and load a communication protocol well suited to current network conditions.

A developer can use reflective services that are either native to a programming language—such as Common Lisp Object System (CLOS), Python, or various Java derivatives—or provided by a middleware platform. When combined with AOP, reflection enables a MOP to weave code for crosscutting concerns into an application at runtime. However, dynamically loading and unloading adaptive code requires the target software modules to exhibit a "plug-and-play" capability.

Component-based design

The third major technology supporting compositional adaptation is component-based design. *Software components* are software units that third parties can independently develop, deploy, and compose.[9] Popular component-based platforms include COM/DCOM, .NET, Enterprise Java Beans, and the Corba Component Model.

Component-based design supports two types of composition. In *static* composition, a developer can combine several components at compile time to produce an application. In *dynamic* composition, the developer can add, remove, or reconfigure components within an application at runtime. To provide dynamic recomposition, a component-based framework must support late binding, which enables coupling of compatible components at run-

Table 1. Example research projects, commercial packages, and standard specifications that provide compositional adaptation.

Project	Institution/Organization
Language-based projects	
AspectJ	Xerox Palo Alto Research Center
Composition filters	Universiteit Twente, The Netherlands
Program Control Language (PCL)	University of Illinois
Open Java	IBM Research
R-Java	University Federal de São Carlos, Brazil
Kava	University of Newcastle, UK
Adaptive Java	Michigan State University
Transparent Reflective Aspect Programming in Java (TRAP/J)	Michigan State University
Middleware-based projects	
Domain-specific services layer:	
Boeing Bold Stroke (BBS)	Boeing
Common services layer:	
CorbaServices	Object Management Group
Quality objects (QuO)	BBN Technologies
Adaptive Corba Template (ACT)	Michigan State University
Interoperable Replication Logic (IRL)	University of Rome, Italy
Distribution layer:	
.NET remoting	Microsoft
Open ORB and Open COM	Lancaster University, UK
The ACE ORB (TAO) and Component Integrated ACE ORB (CIAO)	Distributed Object Computing Group
DynamicTAO and Universally Interoperable Core (UIC)	University of Illinois
Orbix, Orbix/E, and ORBacus	Iona Technologies
Squirrel	University of Kaiserslautern, Germany
AspectIX	Friedrich-Alexander University, Germany
Host infrastructure layer:	
Java virtual machine (JVM)	Sun Microsystems
Common Language Runtime (CLR)	Microsoft
Iguana/J	Trinity College, Dublin
Prose	Swiss Federal Institute of Technology
Adaptive Communication Environment (ACE)	Distributed Object Computing Group
Ensemble	Cornell University
Cross-layer projects	
Distributed Extensible Open Systems (DEOS)	Georgia Institute of Technology
Grace	University of Illinois

time through well-defined interfaces used as contracts. In addition, to provide consistency with other applications, a component-based framework must support coexistence of multiple versions of components.

By enabling the assembly of off-the-shelf components from different vendors, component-based design promotes software reuse. Moreover, mechanisms for maintaining a program's component structure after the initial deployment, when combined with late binding, facilitate compositional adaptation.

Middleware and other factors

In addition to the three main technologies supporting dynamic recomposition, many other factors have contributed to the growth in this area. Perhaps the most important is middleware's increasing role in distributed computing. Middleware provides a layer that developers can exploit to implement adaptive behavior. Indeed, many approaches to compositional adaptation are realized in various middleware layers.

Other technologies important to adaptive software design include software design patterns, mobile agents, generative programming, adaptive programming, and intentional programming.[5]

COMPOSITIONAL ADAPTATION TAXONOMY

Researchers and developers have proposed a wide variety of methods for supporting compositional adaptation. Table 1 lists several research projects, commercial software packages, and standard specifications that support some form of compositional adaptation. The list is by no means exhaustive. Rather, it includes projects that exemplify the

Table 2. Software recomposition techniques.

Technique	Description	Examples
Function pointers	Application execution path is dynamically redirected through modification of function pointers.	Vtables in COM, delegates and events in .NET, callback functions in Corba
Wrappers	Objects are subclassed or encapsulated by other objects (wrappers), enabling the wrapper to control method execution.	ACE, R-Java, PCL, QuO, TRAP/J
Proxies	Surrogates (proxies) are used in place of objects, enabling the surrogate to redirect method calls to different object implementations.	ACT, AspectIX
Strategy pattern	Each algorithm implementation is encapsulated, enabling transparent replacement of one implementation with another.	DynamicTAO and UIC
Virtual component pattern	Component placeholders (virtual components) are inserted into the object graph and replaced as needed during program execution.	ACE and TAO
Metaobject protocol	Mechanisms supporting intercession and introspection enable modification of program behavior.	Open Java, Kava, TRAP/J, Open ORB, Open COM, Iguana/J
Aspect weaving	Code fragments (aspects) that implement a crosscutting concern are woven into an application dynamically.	AspectJ, Composition Filters, TRAP/J, AspectIX, Iguana/J, Prose
Middleware interception	Method calls and responses passing through a middleware layer are intercepted and redirected.	ACT, IRL, Prose
Integrated middleware	An application makes explicit calls to adaptive services provided by a middleware layer.	Adaptive Java, Orbix, Orbix/E, ORBacus, BBS, CIAO, Iguana/J, Ensemble

distinctions in a taxonomy we have developed based on how, when, and where software composition takes place. We have applied the taxonomy to many additional projects.[10]

How to compose

The first dimension of our taxonomy addresses the specific software mechanisms that enable compositional adaptation. Table 2 lists several key techniques with brief descriptions and examples. Mehmet Aksit and Zièd Choukair[8] provide an excellent discussion of such methods.

All of the techniques in Table 2 create a level of indirection in the interactions between program entities. Some techniques use specific software design patterns to realize this indirection, whereas others use AOP, reflection, or both. The two middleware techniques both modify interaction between the application and middleware services, but they differ in the following way: Middleware interception is not visible to the application, whereas integrated middleware provides adaptive services invoked explicitly by the application.

We use the term *composer* to refer to the entity that uses these techniques to adapt an application. The composer might be a human—a software developer or an administrator interacting with a running program through a graphical user interface—or a piece of software—an aspect weaver, a component loader, a runtime system, or a metaobject. Indeed, autonomic computing promises that, increasingly, composers will be software components.

When and where the composer modifies the program determines the *transparency* of the recomposition. Transparency refers to whether an application or system is aware of the "infrastructure" needed for recomposition. For example, a middleware approach to adaptation is transparent with respect to the application source code if the application does not need to be modified to take advantage of the adaptive features. Different degrees of transparency (with respect to application source, virtual machine, middleware source, and so on) determine both the proposed solution's portability across platforms and how easily it can add new adaptive behavior to existing programs.[10]

When to compose

Second, we differentiate approaches according to when the adaptive behavior is composed with the business logic. Generally speaking, later composition time supports more powerful adaptation methods, but it also complicates the problem of ensuring consistency in the adapted program. For example, when composition occurs at development, compile, or load time, dynamism is limited but it is easier to ensure that the adaptation will not produce anomalous behavior. On the other

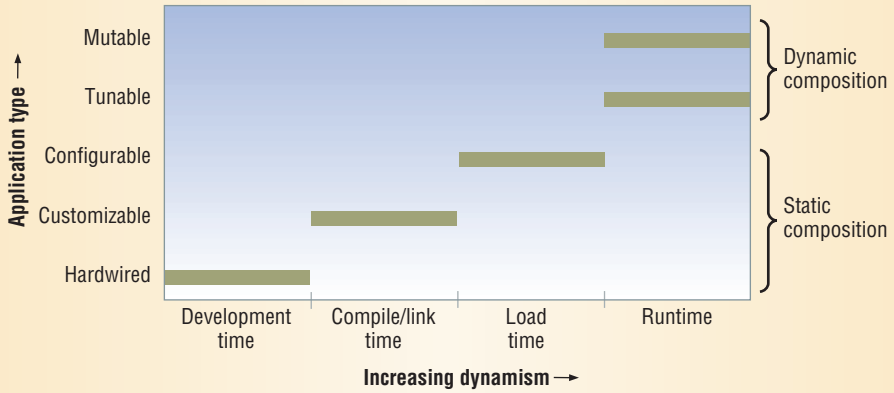


Figure 3. Classification for software composition using the time of composition or recomposition as a classification metric.

hand, while runtime composition is very powerful, it is difficult to use traditional testing and formal verification techniques to check safety and other correctness properties. Figure 3 illustrates the use of composition time as the classification metric for adaptive applications. The vertical axis lists application types that implement either static or dynamic composition. *Static* composition methods take place at development, compile, or load time, whereas *dynamic* composition refers to methods that a composer can apply at runtime.

Static composition. If an adaptive program is composed at development time, then any adaptive behavior is *hardwired* into the program and cannot be changed without recoding.

Alternatively, a developer or user can implement a limited form of adaptation at compile time or link time by configuring the application for a particular environment. For example, aspect-oriented programming languages such as AspectJ enable weaving of aspects into programs during compilation. Aspects might implement an environment-specific security or fault-tolerance policy. Such *customizable* applications require only recompilation or relinking to fit to a new environment.

Configurable applications delay the final decision on the algorithmic units to use in the current environment until a running application loads the corresponding component. For example, the Java virtual machine (JVM) loads classes when a Java application first uses them. Although we consider load-time composition a type of static composition, it offers more dynamism than other static methods. When the application requests the loading of a new component, decision logic might select from a list of components with different capabilities or implementations, choosing the one that most closely matches the current needs. For example, if a user starts an application on a handheld computer, the runtime system might load a minimal display component to guarantee proper presentation.

Other load-time approaches work by dynamically modifying the class itself as it is loaded. For example, to provide runtime monitoring and debugging capabilities, Kava enables the JVM to modify the bytecode as it loads a class.

Dynamic composition. The most flexible approaches to compositional adaptation implement it at runtime. A composer can replace or extend algorithmic and structural units during execution without halting and restarting the program. We differentiate two types of approaches according to whether or not the composer can modify the application's business logic.

Tunable software prohibits modification of code for the business logic. Instead, it supports fine-tuning of crosscutting concerns in response to changing environmental conditions, such as dynamic conditions encountered in mobile computing environments. An example is the fragment object model used in AspectIX, which enables runtime tuning of a Corba application's distribution behavior.

In contrast, *mutable software* allows the composer to change even the program's imperative function, enabling dynamic recomposition of a running program into one that is functionally different. For example, in the OpenORB middleware platform, all objects in the middleware and application code have reflective interfaces, so at runtime the reflective application can change virtually any object in any way, including modifying its interface and internal implementation. While very powerful, in most cases the developer must constrain this flexibility to ensure the system's integrity across adaptations.

Where to compose

The final dimension in which we compare approaches to compositional adaptation centers on where in the system the composer inserts the adaptive code. The possibilities include one of the middleware layers (see Figure A in the "Middleware and Adaptation" sidebar) or the application code itself. In this survey, we do not discuss changes to the operating system; however, we note that operating system extensibility is an active research area. Moreover, adaptations in cross-layer frameworks

Introducing adaptive behavior in higher middleware layers enables portability across virtual machines.

such as DEOS and Grace involve the cooperation of the operating system, middleware, and application.

Middleware layers. Projects involving compositional adaptation at the host-infrastructure middleware layer generally fall in one of two groups. One approach is to construct a layer of adaptable communication services. ACE is an early example that used service wrappers and C++ dynamic binding to support adaptable interprocess communication and event handling services. Ensemble provides a layered architecture that enables a distributed application to select a particular communication protocol.

The second approach is to provide a virtual machine with facilities to intercept and redirect interactions in the functional code. For example, JVM and common language runtime (CLR) facilitate dynamic recomposition through reflection facilities provided by the Java language and .NET platform, respectively. R-Java supports metaobjects by adding a new instruction to the Java interpreter, while Prose and Iguana/J use aspect weaving to add behavioral reflection to the standard JVM. In general, approaches in this category are very flexible with respect to dynamic reconfiguration in that they allow new code to be introduced at runtime. However, they use customized virtual machines to provide transparency to the application, which may reduce portability.

Introducing adaptive behavior in higher middleware layers—distribution, common services, and domain-specific services—enables portability across virtual machines. These approaches typically involve middleware components that intercept messages associated with remote method invocations and redirect or modify them in a manner that accounts for current conditions. For some frameworks, the application developer constructs explicit calls to adaptive middleware services. Examples include Orbix, Orbix/E, ORBacus, CIAO, and Boeing Bold Stroke. QuO uses wrappers around Corba stubs and skeletons to gain control of the call sequence, whereas IRL and ACT use Corba portable interceptors to do so. Portable interceptors serve as "generic" hooks that a composer can use at runtime to load other types of interceptors. Since a user can load a portable interceptor using a command-line parameter, this approach enables the composer to integrate adaptive components into the program without modifying either the application or the middleware code.

Application code. Although middleware approaches support transparent adaptation, they apply only to programs that are written against a specific middleware platform. A more general approach is for developers to implement compositional adaptation in the application program itself.

Two main techniques are available. The first is to program all or part of the application code using a language that directly supports dynamic recomposition. Some languages, such as CLOS or Python, provide support inherently, while others have been extended to support adaptation. For example, Open Java, R-Java, Handi-Wrap, PCL, and Adaptive Java all extend Java to include new keywords and constructs that enhance the adaptive code's expressiveness. However, this approach requires the developer to use these features explicitly in constructing the program.

The second technique is to weave the adaptive code into the functional code. AspectJ and Composition Filters weave adaptive behavior into existing applications at compile time. In contrast, tools such as TRAP/J use a two-step approach to enable dynamic recomposition. In the first step, an aspect weaver inserts generic interception hooks, in this case implemented as aspects, into the application code at compile time. In the second step, a composer dynamically weaves new adaptive components into the application at runtime, and a metaobject protocol uses reflection to forward intercepted operations to the adaptive components. This approach offers a way to add adaptive behavior to existing applications transparently with respect to the original code. Such a capability is important as users expect legacy applications to execute effectively across an increasingly diverse computing infrastructure.

KEY CHALLENGES

Despite many advances in mechanisms to support compositional adaptation, the full potential of dynamically recomposable software systems depends on fundamental advances on four other fronts.

Assurance

Recomposable software design requires a programming paradigm that supports automated checking of both functional and nonfunctional system properties.[11]

To help ensure the adapted system's correctness, developers must first certify all components for correctness with respect to their specifications. They can obtain this certification either by selecting com-

ponents that have already been verified and validated offline using traditional techniques, such as testing, inspection, and model checking, or by generating code automatically from specifications. The certification can include nonfunctional requirements, such as security and performance, as well as functional requirements.

Second, techniques are needed to ensure that the system still executes in an acceptable, or *safe*, manner during the adaptation process. Our group and others are using dependency analysis to address this problem. In addition, developers can use high-level contracts[12] and invariants to monitor system correctness before, during, and after adaptation.

Security

Whereas assurance deals primarily with system integrity, security addresses protection from malicious entities—preventing would-be attackers from exploiting the adaptation mechanisms. In addition to verifying component sources, an adaptive software system must protect its core from attackers. Various well-studied security mechanisms are available, such as strong encryption to ensure the confidentiality and authenticity of messages related to adaptation.

However, the system must also hide adaptation management from would-be intruders and prevent them from impeding or corrupting the adaptation process. A comprehensive approach to this problem must ensure the integrity of the data used in decision-making and conceal the adaptive actions, perhaps by obscuring them within other system activities.

Interoperability

Distributed systems that can adapt to their environment must both adapt individual components and coordinate adaptation across system layers and platforms. Software components are likely to come from different vendors, so the developer may need to integrate different adaptive mechanisms to meet an application's requirements. The problem is complicated by the diversity of adaptive software approaches at different system layers. Even solutions within the same layer are often not compatible.

Developers need tools and methods to integrate the operation of adaptive components across the layers of a single system, among multiple computing systems, and between different adaptive frameworks.

Decision making

Adaptive systems respond to a dynamic physical world. They must act autonomously, modifying software composition to better fit the current environment while preventing damage or loss of service. Decision-making software uses input from software and hardware sensors to decide how, when, and where to adapt the system. Interactive systems may even require the decision maker to learn about and adapt to user behavior.

The system must also hide adaptation management from would-be intruders.

Some researchers have constructed software decision makers using rule-based approaches or control theory. Others have designed decision makers whose actions are inspired by biological processes, such as the human nervous system and emergent behavior in insect species that form colonies.

These approaches to decision making in adaptive software have been effective in certain domains, but environmental dynamics and software complexity have limited their general application. More extensive research in decision making for adaptive software is needed. Future systems must accommodate high-dimensional sensory data, continue to learn from new experience, and take advantage of new adaptations as they become available.

Many of the mechanisms for compositional adaptation are available now, and we expect their use to increase as programmers become more familiar with adaptive software technologies and society comes to expect computer systems to manage themselves. There is a potential downside, however, in the lack of supporting development environments. Compositional adaptation is powerful, but without appropriate tools to automatically generate and verify code, its use can negatively impact—rather than improve—system integrity and security.

The computer science community must build development technologies and tools, well grounded in rigorous software engineering, to support compositional adaptation. This foundation will raise the next generation of computing to new levels of flexibility, autonomy, and maintainability without sacrificing assurance and security. ■

Acknowledgments

We express our gratitude to the many individuals who have contributed to this emerging area of study. Discussions with researchers associated with many of the projects listed in Table 1 have greatly improved our understanding of this area. We also thank the faculty and students in the Software Engineering and Network Systems Laboratory at

Michigan State University for their contributions to RAPIDware, Meridian, and related projects.

This work was supported in part by National Science Foundation grants CCR-9901017, CCR-9912407, EIA-0000433, EIA-0130724, and ITR-0313142, and by the US Department of the Navy, Office of Naval Research, under grant no. N00014-01-1-0744.

Further information

Our group is participating in compositional adaptation research through two projects: RAPIDware (www.cse.msu.edu/rapidware) addresses adaptive software for protecting critical infrastructures, and Meridian (www.cse.msu.edu/meridian) addresses automated software engineering for mobile computing. Among other artifacts, these projects produced ACT, Adaptive Java, and TRAP/J. The technical report on our taxonomy is a "living document" available through the RAPIDware URL.

References

1. M. Weiser, "Hot Topics: Ubiquitous Computing," *Computer*, Oct. 1993, pp. 71-72.
2. J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *Computer*, Jan. 2003, pp. 41-50.
3. G.S. Blair et al., "An Architecture for Next-Generation Middleware," *Proc. IFIP Int'l Conf. Distributed Systems Platforms and Open Distributed Processing* (Middleware 98), Springer, 1998, pp. 191-206.
4. D.L. Parnas, "On the Criteria to Be Used in Decomposing Systems into Modules," *Comm. ACM*, Dec. 1972, pp. 1053-1058.
5. K. Czarnecki and U. Eisenecker, *Generative Programming*, Addison-Wesley, 2000.
6. G. Kiczales et al., "Aspect-Oriented Programming," *Proc. European Conf. Object-Oriented Programming* (ECOOP), LNCS 1241, Springer-Verlag, 1997, pp. 220-242.
7. P. Maes, "Concepts and Experiments in Computational Reflection," *Proc. ACM Conf. Object-Oriented Programming Systems, Languages, and Applications* (OOPSLA), ACM Press, 1987, pp. 147-155.
8. M. Aksit and Z. Choukair, "Dynamic, Adaptive, and Reconfigurable Systems Overview and Prospective Vision," *Proc. 23rd Int'l Conf. Distributed Computing Systems Workshops* (ICDCSW03), IEEE CS Press, May 2003, pp. 84-89.
9. C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed., Addison-Wesley, 2002.
10. P.K. McKinley et al., "A Taxonomy of Compositional Adaptation," tech. report MSU-CSE-04-17, Dept. Computer Science and Engineering, Michigan State Univ., 2004.
11. N. Venkatasubramanian, "Safe 'Composability' of Middleware Services," *Comm. ACM*, June 2002, pp. 49-52.
12. A. Beugnard et al., "Making Components Contract Aware," *Computer*, July 1999, pp. 38-45.

Philip K. McKinley *is a professor in the Department of Computer Science and Engineering at Michigan State University. His research interests include adaptive middleware, mobile computing, pervasive computing, distributed systems, and group communication. McKinley received a PhD in computer science from the University of Illinois at Urbana-Champaign. He is a member of the IEEE Computer Society and the ACM. Contact him at mckinley@cse.msu.edu.*

Sayed Masoud Sadjadi *is a PhD candidate in the Department of Computer Science and Engineering at Michigan State University. His research interests include adaptive software, middleware, pervasive computing, autonomic computing, and sensor networks. Sadjadi received an MS in software engineering from Azad University at Tehran. He is a student member of the IEEE Computer Society and the ACM. Contact him at sadjadis@cse.msu.edu.*

Eric P. Kasten *is a PhD candidate in the Department of Computer Science and Engineering and a software developer in the National Superconducting Cyclotron Laboratory, both at Michigan State University. His research interests include autonomic computing and learning algorithms for adaptable software. Kasten received an MS in computer science from Michigan State University. He is a member of the IEEE Computer Society and the ACM. Contact him at kasten@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 formal methods for software engineering, component-based software development, object-oriented analysis and design, embedded systems development, and visualization. Cheng received a PhD in computer science from the University of Illinois at Urbana-Champaign. She is a senior member of the IEEE Computer Society and a member of the ACM. Contact her at chengb@cse.msu.edu.*

Seamless Mobile Computing on Fixed Infrastructure

Internet Suspend/Resume is a thick-client approach to mobility in which hardware virtualization and file caching are the keys to rapid personalization of anonymous hardware for transient use.

Michael Kozuch
Intel Research Pittsburgh

Mahadev Satyanarayanan
Carnegie Mellon University and Intel Research Pittsburgh

Thomas Bressoud
Denison University

Casey Helfrich
Intel Research Pittsburgh

Shafeeq Sinnamohideen
Carnegie Mellon University

The term "mobile computing" typically evokes images of a laptop, handheld, or wearable computer. However, the plummeting cost of hardware suggests that pervasive computing infrastructure could minimize the need to carry such devices in the near future. We envision a world in which computers are provided for public use in locations ranging from coffee shops to medical office waiting rooms. We can even imagine each seat on an aircraft or a commuter train being equipped with a computer for the convenience of the current occupant.

In such a world, personal computing will be available anywhere on demand, like light at the flip of a switch. Only when a user starts to use a computer will it acquire his unique customization and state. When he finishes using the computer, this customization and state will disappear from it. Thus, a user could travel hands-free yet be confident of making productive use of slivers of free time anywhere. For this to be a compelling vision from a user's viewpoint, the customization and state acquisition process must be accurate and nearly instantaneous. For it to be a viable business model, the management and system administration costs of pervasive deployments of machines must be low.

To address these challenges, we have developed *Internet Suspend/Resume* (ISR), a pervasive computing technology that rapidly personalizes and depersonalizes anonymous hardware for transient use (http://info.pittsburgh.intel-research.net/project/isr). As its name implies, ISR mimics the closing and opening of a laptop. A user can suspend work on an ISR machine at one location, travel to another location, and resume work there on any other ISR machine. Of course, he can also resume on the original ISR machine if he travels with it—this just happens to be an important special case of the more general capability.

Hardware virtualization and file caching are the keys to ISR's precise customization and simple administration. ISR layers *virtual machines* on a location-transparent distributed file system that aggressively caches data. Each VM encapsulates distinct execution and user customization state. The distributed file system transports that state across both space (from suspend site to resume site) and time (from suspend instant to resume instant). Because of its precise state capture and its efficient state transport, ISR is a key enabling technology for pervasive computing.

Our approach eliminates the need for modifications to guest applications or guest operating systems. In particular, ISR supports unmodified Microsoft software including any of the Windows operating systems and the Office application suite. The average user can thus benefit immediately from ISR.

ISR's thick-client approach to mobility is fundamentally different from thin-client approaches. Although thin-client solutions work well in some situations, they are frustrating to use in failure-prone, congested, or high-latency networks because graphical user interactions such as scrolling and highlighting are painful. In contrast, ISR offers crisp interactive performance under all networking conditions, including total disconnection.

0018-9162/04/$20.00 © 2004 IEEE

User Mobility Evolution

Internet Suspend/Resume is the latest step in a long historical evolution toward user mobility on fixed infrastructure. The earliest form of user mobility dates back to the early 1960s and was supported by timesharing systems attached to "dumb" terminals, at any of which users could access their personal environments.

Thin-client approaches such as InfoPad,[1] SLIM,[2] VNC,[3] and xmove[4] are the modern-day realization of this capability, providing just enough compute power to support GUIs. These strategies work well in some situations but are frustrating to use in failure-prone, congested, or high-latency networks. In such systems, even simple user interactions such as scrolling and highlighting can be tedious because current GUI designs assume very low end-to-end latency. ISR's thick-client approach to mobility is fundamentally different, offering crisp interactive performance under all networking conditions, including total disconnection.

Thick-client strategies became possible after the birth of personal computing more than two decades ago. The vision of being able to use any machine as your own, which dates back at least to the mid-1980s, motivated both location transparency and client caching in the Andrew File System. According to a 1990 article on AFS,[5] "A user can walk up to any workstation and access any file in the shared name space. A user's workstation is 'personal' only in the sense that he owns it."

However, AFS only saves and restores persistent state; it does not preserve volatile state such as the size and placement of windows. In addition, the user sees the client's native operating system and application environment, which in many cases may not be the user's preferred environment.

ISR closely resembles process migration. The key difference is that ISR operates as a hardware-level abstraction, while process migration operates as an OS-level abstraction. In principle, ISR would seem to be at a disadvantage because hardware state is much larger. However, in practice, the implementation complexity and software engineering concerns of process migration have proved to be greater practical challenges. Successful implementations of process migration have been demonstrated, but no widely used OS currently supports it as a standard capability.

References

1. T.E. Truman et al., "The InfoPad Multimedia Terminal: A Portable Device for Wireless Information Access," *IEEE Trans. Computers,* vol. 47, no. 10, 1998, pp. 1073-1087.
2. B.K. Schmidt, M.S. Lam, and J.D. Northcutt, "The Interactive Performance of SLIM: A Stateless, Thin-Client Architecture," *Proc. 17th ACM Symp. Operating Systems and Principles,* ACM Press, 1999, pp. 32-47.
3. T. Richardson et al., "Virtual Network Computing," *IEEE Internet Computing,* Jan./Feb. 1998, pp. 33-38.
4. E. Solomita, J. Kempf, and D. Duchamp, "Xmove: A Pseudoserver for X Window Movement," *The X Resource,* Nov. 1994, pp. 143-170.
5. M. Satyanarayanan, "Scalable, Secure, and Highly Available Distributed File Access," *Computer*, May 1990, pp. 9-21.

DESIGN OVERVIEW AND RATIONALE

Simplicity was the driving force behind our decision to use a VM-based approach for state encapsulation. In 2001, Peter Chen and Brian Noble[1] first suggested that a VM's clean encapsulation of state might simplify state migration. In 2002, we reported the first experimental validation of this hypothesis.[2] Since then, a number of academic and commercial efforts—including Zap,[3] work by Constantine Sapuntzakis and colleagues,[4] and GoToMyPC (www.gotomypc.com)—have explored the problem of transferring user customization across machines.

Our implementation uses VMware Workstation (henceforth just "VMware"), a commercial virtual machine monitor (VMM) for the Intel x86 architecture that operates in conjunction with a host operating system and relies on it for services such as device management. VMware runs on several host operating systems and supports a wide range of guest operating systems including Windows 95/98, Windows 2000/XP, and Linux. VMware maps each supported VM's state to host files. After suspension, these files can be copied to another machine with a similar hardware architecture, and VMware can resume execution of the VM there.

Easy deployment

As the "User Mobility Evolution" sidebar describes, ISR bears some resemblance to a system based on process migration. However, a VM-based system is easier to deploy because it better encapsulates volatile execution state. In addition, a VM-based system tolerates greater disparity between the source and target systems across which migration occurs. Successful process migration requires a close match between host and target operating systems, language runtime systems, and so on. In contrast, VM migration only requires a compatible VMM at the target.

If the resume machine is known with certainty at suspend, direct data transfer can be used to transport VM state. However, users may not resume for many hours or days, and the resume machine may not be the one originally anticipated. In the face of such uncertainty, direct data transfer would require the suspend machine to preserve the user's VM file state for an extended period of time. Further, the suspend machine cannot be turned off or unplugged from the network until resume occurs. This violates our goal of granting ISR sites full autonomy to simplify their management—as long as there is no active user at an ISR machine, unplugging it or turning it off should cause no disruption.

These considerations led us to a design in which the Internet is the true home of VM state, and ISR machines are merely temporary usage points of that state. Because VMware stores state in files, a distributed file system is the obvious choice for ISR Internet storage. Distributed file systems are a mature technology, with designs such as the Andrew File System (AFS) that aggressively cache data at clients for performance and scalability.

Using a distributed file system, with each ISR machine configured as a client, is the key to mobility. Demand caching ensures that relevant parts of VM state follow a user from suspend to resume. Using a distributed file system also simplifies site management. Because an ISR machine holds user-specific state only while active, it can be treated like an appliance. The owner of an ISR site can turn an unused machine on or off, move it, or discard it at will without any centralized coordination or notification. The classic client-server model is thus a better match for ISR than a peer-to-peer design, even though it transfers data in two hops—suspend machine to server, then server to resume machine.

Distributed file systems allow a highly asymmetric separation of concerns, thereby reducing the skill needed to administer ISR machines or to deploy new ones. A small professional staff at an operations center can handle tasks that require expertise, such as backup for disaster recovery, load balancing, and adding new users. We expect that an operations center with file servers and professional staff will often be dedicated to a specific company, university, or Internet service provider. In that case, domain-bridging mechanisms such as AFS cells or Kerberos realms will be valuable when users from different organizations use ISR at semipublic locations such as coffee shops or doctor's offices.

Large VM state

A key obstacle to using ISR is *high resume latency*. Today, a typical VM can range in size from a few Gbytes to many tens of Gbytes. Naive approaches to transferring states this large will result in intolerable resume latencies. Fortunately, numerous real-world considerations can be exploited to mitigate these delays.

Temporal locality. User mobility patterns often exhibit temporal locality. For example, consider a common pattern we envision for ISR: A user begins his day by working at home, suspends work and leaves for his office, resumes work at his office, suspends work at the end of the business day, and resumes work at home after dinner. In another example, a corporate campus worker or a factory supervisor might visit coworkers at various locations many times throughout the day, resuming and suspending work at any of those locations.

In these scenarios, caching VM state at fine granularity will translate temporal locality of ISR machine usage into file reference locality at those machines. Resume latency will be much lower at ISR machines with large persistent file caches because misses will occur only on state that has changed since the last use of that machine by its current user.

Distributed file systems reduce the skill needed to administer ISR machines or to deploy new ones.

Proactivity. With the help of higher-level software, it may sometimes be possible to identify likely resume machines and to proactively transfer VM state to those machines, thereby lowering resume latency. Because proactivity merely requires warming a file cache in our design, the consequences of acting on a bad prediction are mild. A bad prediction could evict useful files from a cache and may waste network bandwidth, but there will be no loss of correctness or need for complex cleanup.

State synthesis. Some VM state rarely changes after initialization. For example, installing Windows XP and the Microsoft Office suite on a small VM configuration can consume one-quarter to one-half its virtual disk capacity. Because this state is identical on other similarly configured VMs, it could be captured on read-only media such as CD-ROMs and distributed to ISR machines with poor Internet connectivity. At resume, ISR software could synthesize large parts of the VM state from the read-only media or the file caches of nearby ISR machines, rather than demand-fetching it over a slow network.

Portable storage

USB and FireWire data storage devices are widely available today as unobtrusive flash-memory key chains or microdrives. If a user is willing to carry such a device, the system could copy part of the user's VM state to it at suspend and obtain the state from it at resume, thereby improving performance at poorly connected ISR sites.

However, using a portable storage device presents a number of problems:

- the entire VM state may not fit on the device;
- the copy operation may take too long at suspend for a user who is in a hurry to leave;
- at resume, the VM state on the device could be stale with respect to current VM state—for example, if the user forgets to update the

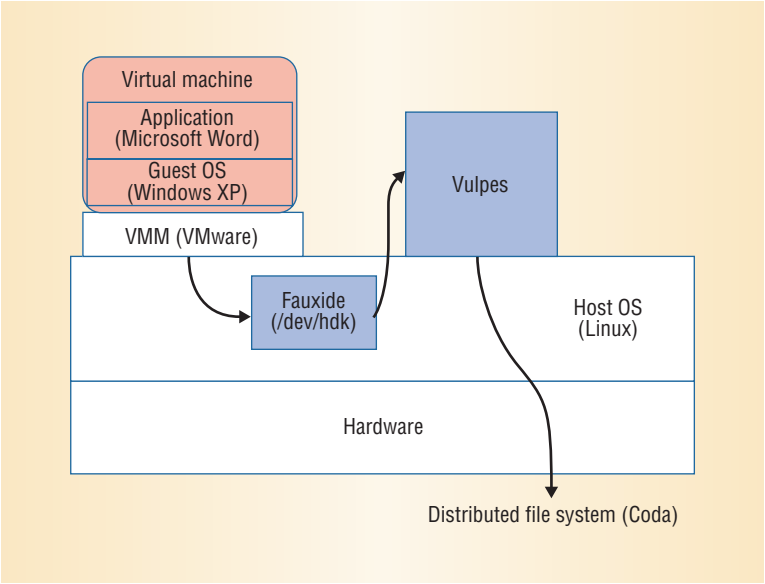


Figure 1. Internet Suspend/Resume host architecture. Fauxide, a loadable kernel module, redirects disk I/O requests on `/dev/hdk` to Vulpes, a user-level process that implements VM state transfer policy and maps VM state to files in Coda.

device at suspend or takes the wrong device when traveling; and
- the device can be lost, broken, or stolen while traveling.

A robust solution must treat portable storage only as a performance assist, not as a substitute, for the underlying distributed file system. Our design makes data on portable devices self-validating: A stale device may not improve performance, but it will never hurt correctness or availability.

ARCHITECTURE AND IMPLEMENTATION

ISR uses the Coda distributed file system[5] for data storage and transport. This choice was based on four key factors:

- complete VM state can fit into a cache because Coda clients cache files on their local disks;
- Coda's support for *hoarding*—anticipatory cache warming—provides a clean interface to exploit advance knowledge of resume machines;
- Coda's support for disconnected and weakly connected operation provides resilience against a wide range of failures and abnormal network conditions; and
- Coda's user-space implementation simplifies experimentation.

ISR represents very large VMware files as a directory tree in Coda rather than as a single file. Virtual disk state is divided into 256-Kbyte chunks, and each chunk is mapped to a separate Coda file. These files are organized as a two-level directory tree to allow efficient lookup and access of each chunk. Our choice of 256 Kbytes is based on a trace-driven analysis of chunk size on performance. The large memory state file is stored in compressed form and uncompressed into `/tmp` just prior to resume.

Figure 1 shows the client architecture that interfaces VMware to Coda. A loadable kernel module, Fauxide, serves as the device driver for a pseudodevice named `/dev/hdk` in Linux. A VM is configured to use this pseudodevice as its sole virtual disk in "raw" mode. Fauxide redirects disk I/O requests to a user-level process, Vulpes, which implements VM state transfer policy. Vulpes also maps VM state to files in Coda and controls the hoarding of those files and their encryption. Because Vulpes is outside the kernel and fully under our control, it is easy to experiment with a wide range of state transfer policies.

Lookaside caching[6] enables a Coda client to use portable devices in a way that reduces vulnerability to human error and device failure. On a cache miss, the client first fetches metadata for the missing object from the server. The Coda metadata definition has been broadened to include the SHA-1 hash of file content. With a valid hash, the client can obtain file content from any matching source. For example, if a mounted portable storage device has a file with matching length and hash, the client can copy it locally rather than fetching the file over a slow network from the file server. Lookaside caching is flexible since misses can be directed to a local file tree, a mounted portable storage device, a nearby NFS or Samba server, a neighboring ISR machine's cache, or even to distributed hash table storage.

Our implementation uses a hash index as a hint to make lookaside caching more efficient. To detect version skew between index and content, Coda recomputes a file's hash after a successful lookaside. In case of a mismatch, Coda redirects the cache miss to the file server. In that case, some small amount of work is wasted on the lookaside path, but consistency is still preserved. Coda's callback mechanism ensures that cached metadata, including the hash, tracks server updates.

EVALUATION

Although our prototype is not yet of production quality, it is robust enough for experiments to evaluate ISR's performance tradeoffs.

From a user's viewpoint, the two dominant questions about ISR performance are "How soon can I begin working?" and "How sluggish is work after I resume?" These questions correspond to the ISR performance metrics *resume latency* and *slowdown*. Ideally, both metrics would be zero. Unfortunately, VM state transfer policies that shrink resume latency may increase slowdown and vice versa. We have therefore conducted a series of controlled experiments to quantify these tradeoffs for typical ISR scenarios.

Experimental methodology

Because ISR is intended for the interactive workloads typical of a laptop environment, we have developed a benchmark that models an interactive Windows user. The Common Desktop Application (CDA) uses Visual Basic scripting to drive Microsoft Office applications such as Word, Excel, PowerPoint, Access, and Internet Explorer. CDA pauses between operations to emulate think time.

Our experimental setup consists of 2.0-GHz Pentium 4 clients connected to a 1.2-GHz Pentium III Xeon server through a 100-Mbps Ethernet. All machines have 1 Gbyte of RAM and run Linux RedHat 7.3. Clients use VMware Workstation 3.1 and have an 8-Gbyte Coda file cache. The VM is configured with 256 Mbytes of RAM and 4 Gbytes of disk storage and runs Windows XP as the guest OS. A NISTNet network emulator controls available bandwidth to servers.

Without ISR support, our setup runs the CDA benchmark in 1,071 seconds. This represents a lower bound on the execution time of any ISR state transfer policy because it eliminates the effects of Fauxide, Vulpes, and Coda. Benchmark time serves as the figure of merit for slowdown.

State transfer policies

Copyout/copyin is the most conservative endpoint in a spectrum of VM state transfer policies. All state is copied out at suspend; resume is blocked until the entire state has arrived. Resume latency can be shortened by three steps, which can be combined to generate a wide range of state transfer policies:

- propagating dirty state to servers before suspend,
- warming the file cache in advance of arrival at the resume machine, and
- letting the user resume before full state has arrived.

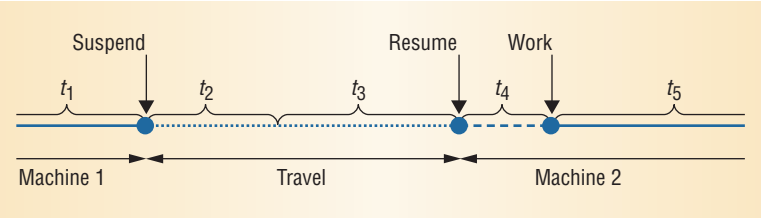


Figure 2. Conceptual ISR timeline. With some state transfer policies, the user can experience significant slowdown during the early part of period t_5.

Figure 2 shows a conceptual timeline for comparing such policies. The figure depicts a user initially working for duration t_1 at machine 1; the user then suspends work and travels to machine 2. In some situations, the system knows (or can guess) machine 2's identity a priori. In other cases, the machine may become apparent only when the user shows up unexpectedly and initiates resume.

The transfer of dirty state from machine 1 to file servers continues after suspend for duration t_2. A period t_3 is then available for proactive file cache warming at machine 2, if known. By the end of t_3, the user has arrived at machine 2 and initiates resume. He experiences resume latency t_4 before he can resume work. He continues working at machine 2 for duration t_5 until he suspends again, and the cycle repeats. With some state transfer policies, the user can experience significant slowdown during the early part of t_5 because some operations block while waiting for the system to transfer missing state.

Baseline policy

The baseline policy is a worst-case straw man. After suspend, all dirty state is transferred to the server during t_2; the period t_3 is empty. At resume, the entire VM state is transferred during t_4 and pinned in the resume machine's cache.

As the "Baseline" column of Table 1 shows, resume latency for this policy is large and highly bandwidth-sensitive. At 100Mbps, resume latency is about 40 minutes. At 10 Mbps, resume latency roughly doubles; it does not increase by a factor of 10 relative to 100 Mbps because Coda, rather than the network, is the bottleneck at the higher bandwidth. Below 10 Mbps, resume latency is intolerable.

The "Baseline" column of Table 2 confirms that slowdown is negligible for the baseline policy. This is because no cache misses occur after resume. Below 100 Mbps, slowdown increases slightly due to Coda background activity such as trickle reintegration.

Table 1. Resume latency, in seconds, for different state transfer policies and bandwidths. Each result is the mean of three trials, with the standard deviation in parentheses.

Bandwidth	Baseline	Fully proactive	Pure demand-fetch	Portable storage lookaside
100 Mbps	2,504 (18)	10.3 (0.1)	14 (0.5)	13 (2.2)
10 Mbps	5,158 (34)	10.2 (0.0)	39 (0.4)	12 (0.5)
1 Mbps	> 9 hours	10.2 (0.0)	317 (0.3)	12 (0.3)
100 Kbps	> 90 hours	11.4 (0.0)	4,301 (0.6)	12 (0.1)

Table 2. Running time, in seconds, of CDA benchmark for different state transfer policies and bandwidths. Each result is the mean of three trials, with the standard deviation in parentheses.

Bandwidth	Baseline	Fully proactive (same as baseline)	Pure demand-fetch	DVD lookaside
100 Mbps	1,105 (9)	←	1,160 (6)	1,141 (36)
10 Mbps	1,170 (47)	←	1,393 (20)	1,186 (17)
1 Mbps	1,272 (65)	←	4,722 (69)	2,128 (34)
100 Kbps	1,409 (38)	←	42,600 (918)	13,967 (131)

Fully proactive policy

If we can predict machine 2, we can define a more aggressive state transfer policy. At machine 2, this policy shifts the entire state transfer time from t_4 to earlier periods in the ISR timeline. During t_3—or earlier, for any state already available at the servers—machine 2 transfers all updated state to its local cache. At resume, all that remains is to launch the VM. This policy is likely to be most effective when a user is working among a small set of sites, such as when alternating between home and work.

As the "Fully proactive" column of Table 1 shows, resume latency is bandwidth-independent and very small (10-11 seconds) because all necessary files are already cached. Post-resume ISR execution is indistinguishable from the baseline. Clearly, this policy is very attractive when feasible.

The minimum travel time needed for a fully proactive policy is $t_2 + t_3$. In the best case, the resume machine is known well in advance, and its cache has been closely tracking the suspend machine. Only the residual dirty state (about 47 Mbytes at the midpoint of the CDA benchmark) must be transferred. This results in a minimum travel time of 45 seconds on a 100-Mbps network and 90 seconds on a 10-Mbps network. These are credible bandwidths and walking distances between collaborating workers at a typical university campus, corporate site, or factory. At 1 Mbps, which is available via DSL or cable modem to many homes today, the best-case travel time is roughly 14 minutes. The attractiveness of ISR in such scenarios is likely to increase over time because networks will improve, but commutes are unlikely to shorten.

Pure demand-fetch policy

Suppose a user arrives unexpectedly at an ISR machine. To keep t_4 short, a pure demand-fetch policy only retrieves memory state during this period. The transfer of VM disk state occurs on demand over t_5, resulting in Coda cache misses that cause slowdown.

For our prototype and benchmark, the state transferred during t_4 is about 41 Mbytes. The time to transfer this state is a lower bound on resume latency for this policy. As the "Pure demand-fetch" column of Table 1 shows, resume latency increases from under one minute at LAN speeds to more than one hour at 100 Kbps.

The slowdown for a pure demand-fetch policy is highly sensitive to bandwidth, as Table 2 shows. The total benchmark time increases from 1,105 seconds without ISR to 1,160 seconds at 100 Mbps; this represents a slowdown of about 8.3 percent. As bandwidth drops, the slowdown increases to 30.1 percent at 10 Mbps, 340.9 percent at 1 Mbps, and well over an order of magnitude at 100 Kbps. Although slowdowns below 100 Mbps will undoubtedly be noticeable, their impact must be balanced against the improvement in user productivity resulting from the ability to work anywhere, even at unexpected locations.

Demand-fetch with lookaside policy

Lookaside caching can improve the pure demand-fetch policy in many ways. If a user is willing to wait briefly at suspend, the VM memory state file can be written to a portable storage device. At the resume machine, lookaside caching from the device can reduce t_4. If read-only or read-write media with partial VM state are available at the resume machine, lookaside caching can use them to reduce the cost of cache misses during t_5.

Table 1 presents results for the first of these scenarios. A comparison of the "Pure demand-fetch" and "Portable storage lookaside" columns of this table show a noticeable improvement below 100

Mbps and a dramatic improvement at 100 Kbps. A resume time of just 12 seconds rather than 317 seconds (at 1 Mbps) or 4,301 seconds (at 100 Kbps) can make a world of difference to a user who only has a few minutes of time available to work while in a coffee shop or a waiting room.

To explore the impact of lookaside caching on slowdown, we used a DVD as a lookaside device. The DVD contained VM state captured after installation of Windows XP and the Microsoft Office suite but before any user-specific or benchmark-specific customizations. Roughly half of the file cache misses are satisfied through lookaside. As Table 2 shows, using lookaside consistently reduces benchmark time relative to a pure demand-fetch policy, particularly at lower bandwidths.

Seamless ubiquitous access to a user's uniquely customized personal environment is the holy grail of mobile computing. ISR represents an important step toward this goal. By exploiting advance knowledge of travel, transferring VM state incrementally, and using portable storage, we have shown that the performance cost of seamless mobility can be made acceptable. Using these techniques, resume latency in ISR is little more than the typical delay a user experiences when opening a laptop. By leveraging the consistency of a distributed file system, ISR is robust in the face of many human errors and is tolerant of poorly managed environments.

ISR is a versatile mechanism with value beyond mobile computing. By severing the tight binding between personal computing state and computer hardware, ISR frees PC state for use in many innovative ways. For example, replicas of PC state could be saved at widely separated Internet sites to allow disaster-recovery after catastrophic failures. As another example, a time-stamped snapshot of PC state could be created, encrypted, digitally signed, and asynchronously transmitted to a verification authority to demonstrate compliance with security advisories. In both cases, a user's foreground activity can continue with minimal disruption.

In the future, we plan to explore these and other exciting opportunities that ISR makes possible. We also expect to address security concerns: Users must be confident that anonymous computers are safe for them to use. We currently assume that deployed machines are secure, but we are planning research to enforce this assumption. To address this difficult challenge, we expect to leverage previous work on secure coprocessors[7,8] and ongoing research by the Trusted Computing Group (www.trustedcomputinggroup.org) and Microsoft's Next-Generation Secure Computing Base (www.microsoft.com/resources/ngscb/default.mspx). ■

References

1. P.M. Chen and B.D. Noble, "When Virtual Is Better Than Real," *Proc. 8th Workshop Hot Topics in Operating Systems,* IEEE CS Press, May 2001, pp. 133-138.
2. M. Kozuch and M. Satyanarayanan, "Internet Suspend/Resume," *Proc. 4th IEEE Workshop Mobile Computing Systems and Applications,* IEEE CS Press, June 2002, pp. 40-46.
3. S. Osman et al., "The Design and Implementation of Zap: A System for Migrating Computing Environments," *Proc. 5th Symp. Operating Systems Design and Implementation,* ACM Press, Dec. 2002, pp. 361-376.
4. C.P. Sapuntzakis et al., "Optimizing the Migration of Virtual Computers," *Proc. 5th Symp. Operating Systems Design and Implementation,* ACM Press, Dec. 2002, pp. 377-390.
5. M. Satyanarayanan, "The Evolution of Coda," *ACM Trans. Computer Systems,* May 2002, pp. 85-124.
6. N. Tolia et al., "Integrating Portable and Distributed Storage," *Proc. 3rd Usenix Conf. File and Storage Technologies,* Usenix Assoc., Mar. 2004, pp. 227-238.
7. S.W. Smith and V. Austel, "Trusting Trusted Hardware: Towards a Formal Model for Programmable Secure Coprocessors," *Proc. 3rd Usenix Workshop Electronic Commerce,* Usenix Assoc., 1998, pp. 83-98.
8. J.D. Tygar and B. Yee, "Dyad: A System for Using Physically Secure Coprocessors," *Proc. Joint Harvard-MIT Workshop Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment,* 1993; www.cni.org/docs/ima.ip-workshop/Tygar.Yee.html.

Michael Kozuch *is a senior researcher at Intel Research Pittsburgh, where his focus is on novel uses of virtual machine technology. Kozuch received a PhD in electrical engineering from Princeton University. Contact him at michael.a.kozuch@intel.com.*

Mahadev Satyanarayanan *is the Carnegie Group Professor of Computer Science at Carnegie Mellon University and the founding director of Intel Research Pittsburgh. His research interests span mobile computing, pervasive computing, and dis-*

tributed systems. Satyanarayanan received a PhD in computer science from Carnegie Mellon University. He is a member of the IEEE Computer Society, a Fellow of the ACM and the IEEE, and the founding editor in chief of IEEE Pervasive Computing. *Contact him at satya@cs.cmu.edu.*

Thomas Bressoud *is an assistant professor of computer science in the Department of Mathematics and Computer Science at Denison University and an affiliate researcher at Intel Research Pittsburgh. His research interests include fault tolerance, content-addressable storage, and distributed systems. Bressoud received a PhD in computer science from Cornell University. Contact him at bressoud@denison.edu.*

Casey Helfrich *is a research engineer at Intel Research Pittsburgh. His research interests include distributed systems, virtualization of hardware, and building research systems. Helfrich received a BS in computer science from Carnegie Mellon University. Contact him at casey.j.helfrich@intel.com.*

Shafeeq Sinnamohideen *is a PhD student in the Computer Science Department at Carnegie Mellon University working on the Internet Suspend/Resume project at Intel Research Pittsburgh. His main research interest is in wide-area distributed file systems. Sinnamohideen received an MS in electrical and computer engineering from Carnegie Mellon University. Contact him at shafeeq+www@cyrus.watson.org.*

Policy-Based Dynamic Reconfiguration of Mobile-Code Applications

Mobility complicates application programming by requiring developers to define when and where to move which components under varying operating conditions. A policy-based implementation framework supports high-level reconfiguration strategies that separate mobility concerns from application functionality.

Rebecca Montanari
University of Bologna

Emil Lupu
Imperial College London

Cesare Stefanelli
University of Ferrara

Global wireless networks have opened the way to a ubiquitous Internet computing environment in which a variety of portable devices remain connected to the Internet fabric—even as their locations change continuously—and access to data and services is independent of both device type and location. In this environment, applications must be able to change the location of their execution. In addition, they must allow heterogeneous and resource-limited portable devices to download only the device-specific software components needed for execution.

Code mobility enables dynamic customization and configuration of ubiquitous Internet applications.[1] Mobile applications can transfer the execution of software components from one device to another depending on resource availability. They can also adapt functionality according to user needs and device characteristics.

Developers are using programming paradigms and techniques based on code mobility in several application domains, ranging from distributed information retrieval and computer-supported cooperative work to network management and mobile user and terminal support.[2-4] However, current approaches force application programmers to explicitly define and control the reconfiguration strategies that determine which software components to move, under what circumstances, and where. Typically, the strategies are embedded in the code in a tightly coupled way that complicates both the application's design and its runtime adaptation to changes in the execution environment.

We have developed a policy-based approach to mobility programming that expresses and controls reconfiguration strategies at a high level of abstraction, separate from the application's functionality. The Policy-Enabled Mobile Applications (Poema) framework provides an integrated environment for developing applications that can change both their functionality and layout at runtime in response to environment conditions. The software is available for download at www.lia.deis.unibo.it/Research/Poema/.

MOBILE-CODE PROGRAMMING

Current mobile-code programming models try to solve Internet-based application design and deployment issues by making location a first-class design concept that programmers can use to control application behavior and layout. The models differ in how they distribute application data and code:[1]

- *remote evaluation* models enable a computational component to send code to another node

0018-9162/04/$20.00 © 2004 IEEE

Related Work in Dynamic Mobility Management

The following mobile-code programming models all share the principle of separating mobility from computational concerns as a key design requirement. However, the models differ on how to achieve this separation.

FarGo[1] provides methods for dividing the application code into mobile modules, called *complets*, and allows separate specification of how to relocate the modules. Programmers use a scripting language to encode complet layout strategies and bind them to the application at loading time.

The MAGE model[2] introduces the abstraction of mobility attributes to describe the mobility semantics of application components. As with FarGo, programmers bind the mobility attributes to the application components at loading time.

Another approach[3] focuses on the mobile agent paradigm and suggests separately encoding an application's function, mobility, and management.

XMILE[4] supports fine-grained mobility, allowing even individual lines of code to be sent across the network, and exploits XML to program mobility patterns.

Some other approaches focus on algorithms and mechanisms for carrying out the runtime reconfiguration of mobile-code applications. For instance, Dacia[5] proposes several mechanisms for consistently changing the application structure at runtime according to the reconfiguration strategies embedded into application-specific modules at design time.

Poema differs from all these approaches in its ability to specify mobility strategies at a higher level of abstraction and to support their modification even during application execution. In addition, Poema's infrastructure offers middleware services that provide support for the design, development, and deployment of adaptive applications in various scenarios.

References

1. O. Holder, I. Ben-Shaul, and H. Gazit, "Dynamic Layout of Distributed Applications in FarGo," *Proc. 21st Int'l Conf. Software Eng.*, ACM Press, 1999, pp. 163-173.
2. E. Barr, R. Pandey, and M. Haungs, "MAGE: A Distributed Programming Model," *Proc. 21st Int'l Conf. Distributed Computing*, IEEE CS Press, 2001, pp. 303-312.
3. K.J. Lauvset, D. Johansen, and K. Marzullo, "Factoring Mobile Agents," *Proc. Workshop on Eng. Computer-Based Systems*, IEEE CS Press, 2002, pp. 253-257.
4. C. Mascolo, L. Zanolin, and W. Emmerich, "XMILE: An XML-based Approach for Incremental Code Mobility and Update," *Automated Software Eng. J.*, Apr. 2002, pp. 151-165.
5. R. Litiu and A. Prakash, "Developing Adaptive Groupware Applications Using a Mobile Component Framework," *Proc. 2000 ACM Conf. Computer-Supported Cooperative Work*, ACM Press, 2000, pp. 107-116.

for remote execution,
- *code on demand* models allow a computational component to load some code from a remote repository when needed, and
- *mobile agent* models migrate an entire computational entity (code and execution state) to a different node.

Most existing mobile-code programming environments support only a single mobility model. As a result, designers can only partially exploit the power of code mobility. In addition, most environments offer only basic runtime support for code mobility and limited programming abstractions and tools for specifying reconfiguration requirements.

The "Related Work in Dynamic Mobility Management" sidebar describes some other programming development models that, like Poema, separate application logic and reconfiguration concerns. However, these models do not support high-level specification of mobility strategies or modification of the strategies during the application execution.

POLICY-BASED PROGRAMMING

Policies are declarative rules governing choices in a system's behavior.[5] They are increasingly popular in both academia and industry as a means for constraining system behavior. Network and system management tools have relied upon policy-based techniques for several years to add flexibility and adaptability to management infrastructures.

Our work extends policy-based techniques to dynamic reconfiguration of mobile-code applications. Developers and even application administrators can dynamically load or remove policies, thus reconfiguring the application at runtime without requiring application code changes. Moreover, policies favor application reusability in different deployment scenarios. Policy templates can be used to encode common reconfiguration strategies and thus facilitate their reuse.

A policy-based mobile-code programming environment requires a policy-specification model to express reconfiguration strategies. Multiple specification approaches are possible including formal policy languages, rule-based policy notations, and attribute table representations.[5]

For dynamic reconfiguration, we advocate using event-triggered declarative rules. Three main reasons underpin this choice:

- an event-driven model is a natural candidate for notifying distributed components of changes that occur in both application and system state;
- declarative policies let programmers directly express what reconfiguration is needed without having to specify how to achieve it, thus facilitating the definition of reconfiguration strategies; and
- declarative policy specifications are amenable to policy analysis and verification. Unlike policies buried in implementation code, declarative policies can be validated externally—for example, via theorem-provers—as sufficient and correct for the application reconfiguration.

Using policies also requires middleware support for policy lifecycle management. In particular, the middleware must provide the management operations needed to compile high-level policies into enforceable representations, such as Java bytecode. It must also distribute policies to interested components and enforce them when an event occurs. Other middleware services should detect and resolve conflicts between policies when new policies are added or removed at runtime.

APPLICATION RECONFIGURATION WITH POEMA

The Poema policy-based framework evolved from our past work on dynamic control of mobile agents.[6] The framework permits the specification of different reconfiguration patterns at different granularities of code mobility, and it supports runtime application reconfiguration.

Programming reconfiguration policies

A Poema application is implemented in two phases. In the *application programming phase*, developers define and code only the application functionality without addressing reconfiguration issues. In a separate *policy specification phase*, they specify the desired reconfiguration strategies.

Poema specifies reconfiguration strategies in terms of Ponder *obligation policies*.[7] These policies are declarative event-condition-action rules that Poema uses to define the desired application reconfiguration when an event occurs.

To illustrate the use of Poema policies, we introduce a personal assistant application that supports the cooperative work of mobile users equipped with Internet-enabled portable devices. The application client module lets users view and update the shared information that the server module manages and maintains. Policies permit the specification of changes in the client and server module configurations and selection of the most appropriate mobility model—for example, code-on-demand to enable the client module to load a new image viewer.

Policy specification. Figure 1 shows the obligation policies for reconfiguring the personal assistant application. In the obligation policy type in Figure 1a, the `on` clause identifies the triggering event, which can be either *application-dependent*, generated by changes in application state, or *application-independent*, indicating a change in the execution environment. Event expressions can also combine events denoting the occurrence of specific event combinations.

The `subject` identifies the *computational components*—the autonomous units of execution that initiate a reconfiguration action. The `target` identifies the CCs to which the reconfiguration actions apply.

In the case of remote evaluation and code-on-demand, subject CCs ship or fetch code respectively to or from target CCs. In the case of mobile agents,

(a) Generic policy type

```
inst oblig policyName "{"
on          event-specification;
subject     domain-Scope-Expression;
target      domain-Scope-Expression;
do          obligation-action-list;
when        constraint-Expression;
"}"
```

(b) Personal assistant policy instances

```
domain a = /PoemaOrganisationID/DomainID;
inst oblig UserMobility_P1 {
on          ChangeDevice(userName, deviceName);
subject     s = a/siteA/UserProxyID;
target      t = a/siteA/UserProxyID;
do          t.go((deviceName.getSite()).toString(), "run()");
when        MonitoringSystem.checkResStatus(deviceName) = = true;
}
```

```
domain a = /PoemaOrganisationID/DomainID;
inst oblig LoadImageViewer_P1 {
on          MissingCode("ImageViewer");
subject     s = a/siteA/UserProxyID;
target      t = a/siteA/UserProxyID;
do          t.fetchCode("/PoemaOrganisationID/DomainID/siteRepository/CodeProvider",
            "ImageViewer");
}
```

```
domain a = /PoemaOrganisationID/DomainID;
inst oblig BatteryShortage_P1 {
on          BatteryStatus(deviceName, BELOW_THRESHOLD);
subject     s = a/sitePDA/SYSTEMCC;
target      t = a/siteA/UserProxyID;
do          t.relocate("/PoemaOrganisationID/DomainID/siteHostA",
            "run()");
}
```

(c) Personal assistant policy types

```
domain a = /PoemaOrganisationID/DomainID;
type oblig UserMobility (subject s, target t) {
on          ChangeDevice(userName, deviceName);
do          t.go((deviceName.getSite()).toString(), "run()");
when        MonitoringSystem.checkResStatus(deviceName) = = true;
}
inst oblig UserMobility_P1 = UserMobility(a/siteA/UserProxyID, a/siteA/UserProxyID)
```

```
domain a = /PoemaOrganisationID/DomainID;
type oblig LoadImageViewer (subject s, target t) {
on          MissingCode("ImageViewer")
do          t.fetchCode("/PoemaOrganisationID/DomainID/siteRepository/CodeProvider",
            "ImageViewer");
}
inst oblig LoadImageViewer_P1 = LoadImageViewer(a/siteA/UserProxyID, a/siteA/UserProxyID)
```

```
domain a = /PoemaOrganisationID/DomainID;
type oblig BatteryShortage (subject s, target t) {
on          BatteryStatus(deviceName, BELOW_THRESHOLD);
do          t.relocate("/PoemaOrganisationID/DomainID/siteHostA",
            "run()");
}
inst oblig BatteryShortage_P1 = BatteryShortage(a/sitePDA/SystemCC, a/siteA/UserProxyID)
```

Figure 1. Obligation policies examples. (a) Generic policy type, (b) policy instances, and (c) policy types for the dynamic reconfiguration of the personal assistant application.

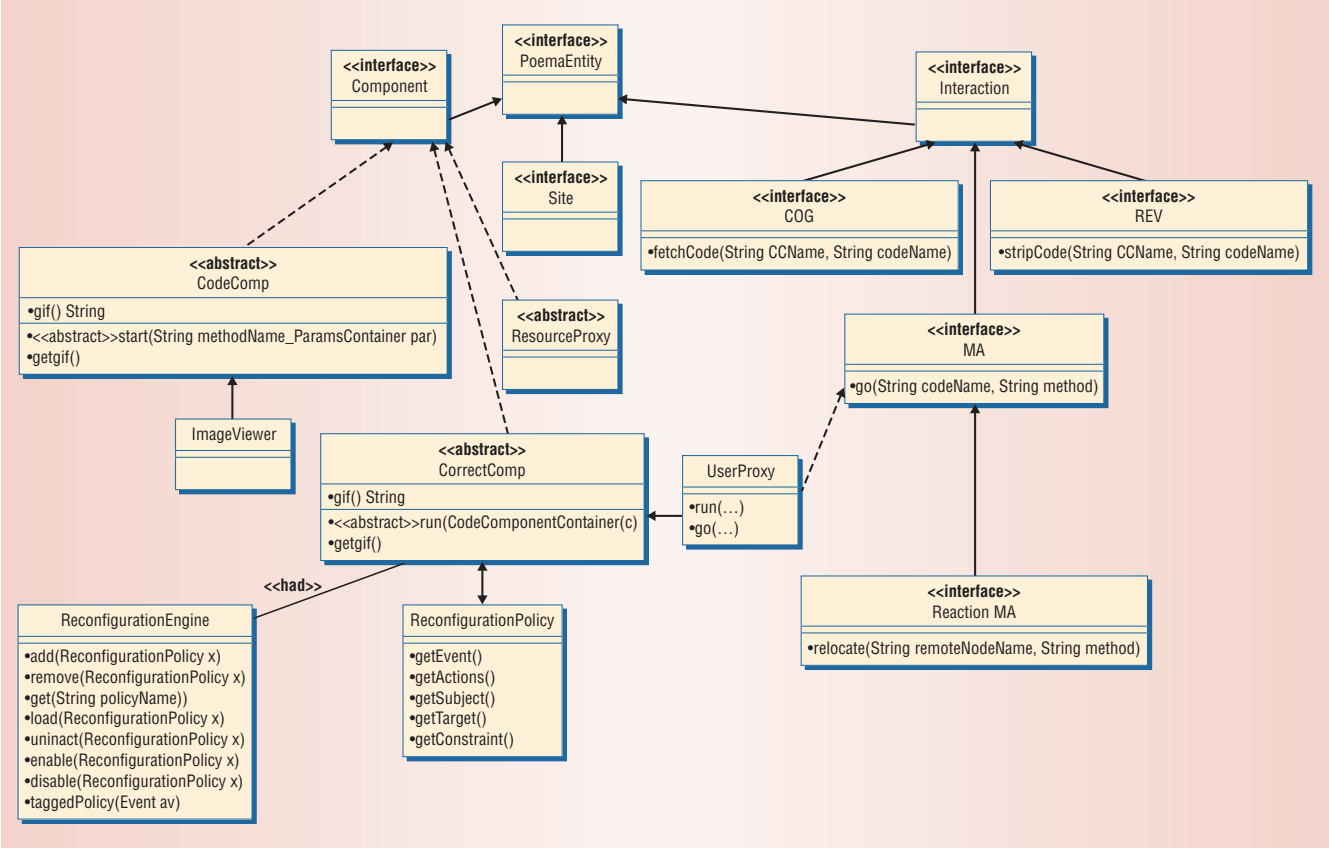


Figure 2. Unified Modeling Language diagram of the main Poema application architecture abstractions.

subject CCs either autonomously migrate or, if authorized, force target CCs to migrate.

The `do` clause identifies the reconfiguration action to perform when the event occurs. The Ponder language includes sequence and concurrency operators that support combinations of various reconfiguration actions.

The `when` clause defines the conditions that must hold for the policy to be applied. Developers can specify these conditions in terms of both application state and environment variables.

Figure 1b shows three policy instances used in the personal assistant application: UserMobility_P1, LoadImageViewer_P1, and BatteryShortage_P1. Note that policy subjects and targets are systemwide and identified by a globally unique name that complies with the Internet's lightweight directory access protocol. Names are a concatenation of the site ID where the subjects and targets were first created with a unique identifier within that site.

When the user changes the access device (the ChangeDevice() event), the UserMobility_P1 policy instance specifies that the client module (/siteA/UserProxyID) must migrate toward the new device if it has sufficient resources to support execution. An underlying monitoring system verifies resource availability. This is a *proactive* mobile agent strategy because the client module autonomously decides to move together with the current session state.

The LoadImageViewer_P1 policy instance in Figure 1b automatically loads a suitable image viewer in the client module from the code provider (/siteRepository/CodeProvider) as needed. The MissingCode() event triggers this policy that models a code-on-demand reconfiguration pattern.

The BatteryShortage_P1 policy instance in this figure takes into account the possible battery power degradation in the portable device where the client module is currently executing (sitePDA).

In this case, a system-level computational component (SystemCC) on sitePDA relocates the client module to the node siteHostA, saving the current session state and permitting the client module to continue its execution. This policy implements a *reactive* mobile agent strategy in which an external entity forces the client module to move.

The Ponder language offers several benefits for reconfiguration. Its event-triggered obligation policies simplify the programming of CC reactions to changing conditions. Simply modifying the specification of either the event or the action causes CCs to react differently without changing their code.

In addition, the explicit distinction between policy subject and target permits modeling both proactive and reactive CC migrations. In proactive migration, the subject coincides with the target, as in the UserMobility_P1 policy. In reactive migration, the subject differs from the target, as the BatteryShortage_P1 policy.

Policy reuse. Ponder supports policy reusability in different application scenarios through the adoption of policy types. As Figure 1c shows, programmers

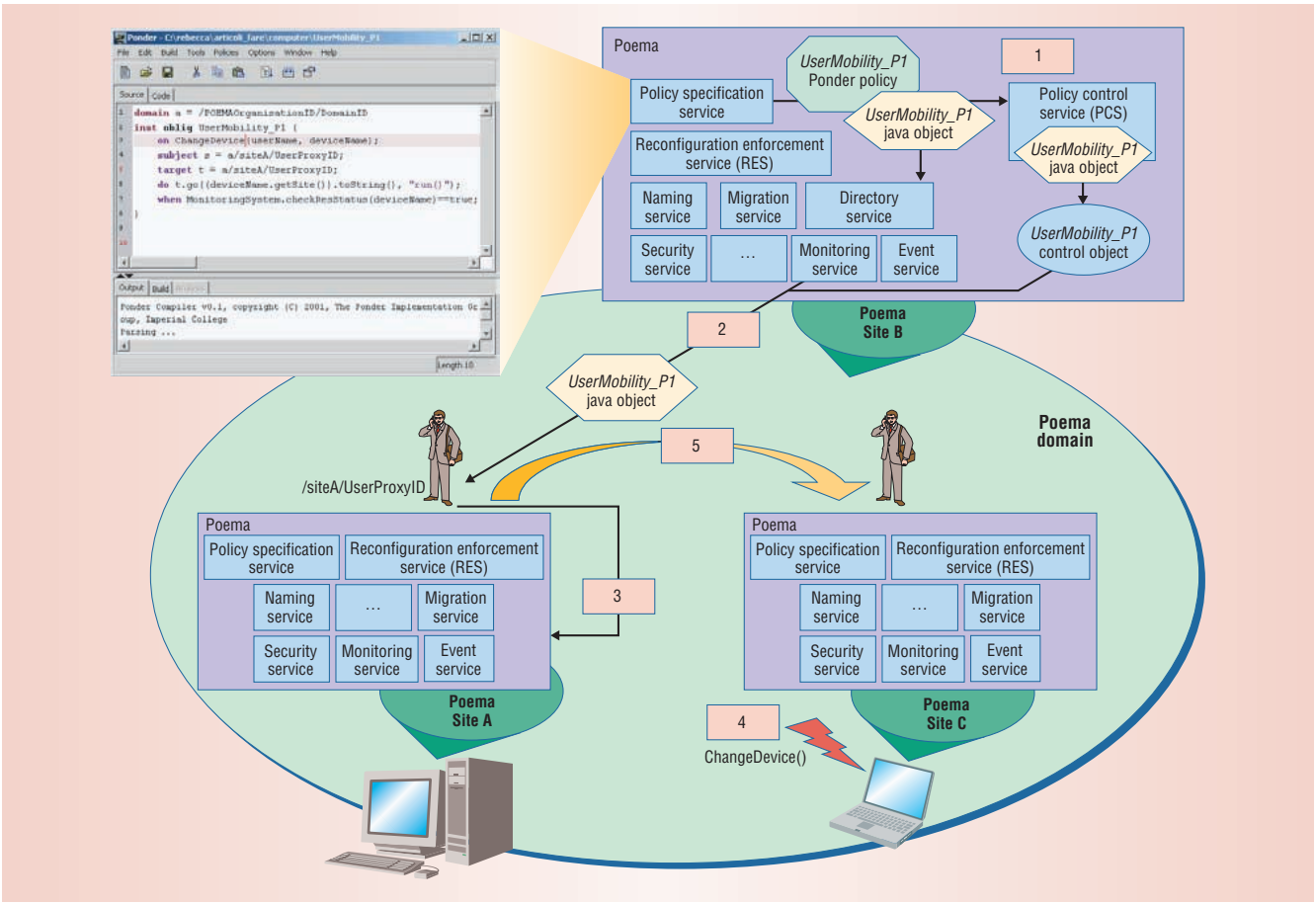


Figure 3. Poema middleware. Schematic shows the deployment of a policy-based personal assistant application in one domain.

can exploit policy types to introduce specialized user-defined mobility patterns. They can parameterize policy types and create policy instances with application-specific parameters, such as subjects and targets.

Policy conflicts. Conflicts can arise because several policies may apply to the same CC. Previous work on the Ponder language distinguishes between *modality conflicts*—conflicts between permissions and prohibitions or between obligations and prohibitions—and *application-specific conflicts*.[8] More recent work on policy analysis in Ponder has focused on using an event-calculus-based policy representation together with abductive reasoning to identify the conditions and scenarios that lead to a conflict.[9]

Poema application architecture

Poema models an application in terms of components allocated in different sites and cooperating via interaction patterns. Figure 2 shows the main abstractions in the application architecture, which includes several *components*:

- CCs (ComputComp class) represent autonomous execution flows,
- code components (CodeComp class) define application-specific functionality, and
- resource components (ResourceProxy class) identify logical/physical resources.

Sites typically model network nodes that host components and provide execution environments for CCs. *Domains* are sites grouped to correspond to network localities, such as Ethernet-based LANs, and possibly organized in a hierarchy. Poema treats remote-evaluation, code-on-demand, and mobile-agent paradigms as specific *interaction* patterns. It provides the methods for sending and receiving code components as well as moving the entire CC either proactively or reactively (respectively, `shipCode()`, `fetchCode()`, `go()`/`relocate()`).

Developers encode any CC's application logic flow in the `run()` method and implement it as call sequences to the code components' `start()` methods. The type and number of code components that Poema executes depend on the application goals.

Note that the `run()` method does not contain reconfiguration directives, which are instead embedded into the reconfiguration policies (ReconfigurationPolicy class). Any CC contains the references to its policy objects. Poema provides all the needed policy management support—for example, policy load/unload and reconfiguration activation (ReconfigurationEngine class).

POEMA MIDDLEWARE

Figure 3 shows the two most important sets of

Poema exploits domain abstractions to facilitate policy lifecycle management and to improve scalability.

reconfiguration support services that Poema provides:

- the *policy* set contains specific services for the runtime deployment of Ponder policies; and
- the *basic* set provides core support services, such as naming, migration, monitoring, event registration/dispatching, and security.

Policy services

The *policy specification service* provides tools for defining browsing and editing policies and for initializing reconfiguration policy objects. Initialization consists of generating a Java policy object that the runtime environment can load and interpret and storing both the Ponder and Java policy representations in a directory service.

The *policy control service* (PCS) is responsible for policy activation. It receives the Java representation of any new policy and generates a control object that keeps track of policy status transitions (initialized, activated, and removed). The control object interrogates the naming service to retrieve the current location of CCs interested in the policy and distributes the policy accordingly.

When the application first instantiates a CC, the PCS instructs the control objects to distribute the policies that apply to it. When a policy is deleted, the PCS coordinates with the responsible control object to inform the interested CCs to unload the policy and then removes the object. Note that Poema represents any policy change, such as policy creation or deletion, and any modification of the CC status, such as CC instantiation or termination, in terms of events that the event service communicates to the PCS.

The *reconfiguration enforcement service* (RES) provides the mechanisms for verifying policy applicability and enforcing reconfiguration policies at runtime. When an event occurs, RES retrieves all triggered policies, interprets them, and executes the needed reconfiguration actions according to policy specifications. If no policy applies, RES enforces specific exception-handling policies.

Reconfiguration can, however, interfere with a component's execution. For example, an asynchronous event might require CC migration at any execution point, possibly violating the consistency of resource bindings. In the current RES implementation, the CC reconfiguration can take place only outside sections the application programmer has explicitly labeled critical.

Basic service set

Figure 3 shows the most important infrastructure services that the Poema middleware provides. The *naming service* associates the CC with a globally unique identifier. CCs can exploit the *migration service* to either migrate among sites or to fetch or ship code respectively from or to other CCs.

The *monitoring service* and *event service* inform CCs about relevant state changes at both the application and operating environment levels. CCs can register their interest in one or more specific events. The event service dispatches event notifications to interested entities even when they have migrated to another location.

The *security service* enforces access controls according to authorization policy specifications. Poema exploits Ponder authorization policies for defining who can access, distribute, and enforce reconfiguration policies and under what circumstances.[6] For instance, in the personal assistant application, only trusted users with the required permissions can access the BatteryShortage_P1 policy stored in the directory. In addition, the /sitePDA/SystemCC component loads the BatteryShortage_P1 policy only if it trusts the policy supplier, and it enforces the client module migration only if the component has the necessary relocation privilege.

Poema integrates trust management solutions based on public-key infrastructure technologies to collect and verify the information required in trust decisions. In particular, Poema entities (programmers, administrators, the PCS, and so on) are associated with digital certificates for authenticating entities, signing policies, and securing communications.

Finally, Poema exploits domain abstractions to facilitate policy lifecycle management and to improve the scalability of policy-based mobile-code application deployment in large-scale environments. Domains define specific policy management boundaries: Each domain holds references to its CC members and the reconfiguration policies that currently apply to them. Applications deployed across several domains require tight interdomain coordination between Poema middleware services. Domains managed by different administrative authorities require a trust infrastructure.

PERSONAL ASSISTANT IMPLEMENTATION

In the personal assistant application in Figure 3, the administrator tool on site B defines the UserMobility_P1 policy. The tool provides predefined obligation policy templates to fill in application-specific data such as subjects, targets, and trigger-

ing events. This approach simplifies policy specification. The Ponder compiler, which is part of the administrator tool, generates the Java policy object—in this case, an instance of the ReconfigurationPolicy class from Figure 2.

The PCS then instantiates a specific UserMobility_P1 control object in charge of sending the Java policy object to the /siteA/UserProxyID client module. If the client module is temporarily unreachable, the control object suspends policy installation until the monitoring service detects its reconnection. The client module loads the received policy and registers the ChangeDevice() event with the event service. The client module can then migrate to any new access device where the user logs in.

Dynamic reconfiguration

At runtime, when the user connects to site C, the event service delivers the ChangeDevice() event to the client module. The client module then delegates the RES to perform the necessary reconfiguration actions via the triggerPolicy() method of the ReconfigurationEngine class shown in Figure 2. RES parses the UserMobility_P1 policy object to determine the subject, target, and other policy elements and enforces the client module migration to site C. If the new device is too limited to host the client module execution, RES handles the exception by sending the user a warning message that it cannot move the client module until the user connects to a more powerful device.

Poema also manages runtime policy removal. For instance, to delete the UserMobility_P1 policy, the PCS forces the UserMobility_P1 control object to contact the client module. The client module deletes the policy from its set of loaded policies, and the control object deregisters the ChangeDevice() event.

Poema provides several policy distribution strategies, depending on application requirements. At one extreme, the client module loads all the Java policy objects that control its reconfiguration. This favors prompt reaction to event notification but increases the client module size and migration time. At the other extreme, the client module loads only the references to its Java policy objects and retrieves the policies on demand from the directory when relevant events occur. Any intermediate solution is also possible.

Multiple concurrent events can lead to inconsistencies in the client module computation. To avoid this problem, RES adopts a sequential approach to policy enforcement based on the order in which the client module receives events. In addition, RES suspends reconfiguration while the client module is inside a critical section of code.

Security

To address policy-deployment security concerns, the personal assistant prototype trusts only the application administrator to define, modify, or remove policies. At policy specification time, the administrator digitally signs the policies. The client module loads only signed policies. The application's computational components—for example, the client and server modules—and the Poema middleware services—the PCS and event service—communicate through secure channels. Using digital signatures in conjunction with secure channels gives the client module evidence of the policy's author (the administrator) and ensures policy integrity.

Poema provides several policy distribution strategies, depending on application requirements.

Policy management

The Poema middleware configuration must define the initial service allocation on the domain's sites. For instance, Figure 3 features one directory service and one PCS per domain, while all other services, such as the RES, are allocated on every site.

A single directory simplifies policy analysis. To verify the consistency of any new policy, the Poema policy conflict checker analyzes all the policies stored in the directory. A single PCS centralizes policy activation, thus facilitating policy distribution and the management of concurrent and possibly conflicting policy activation and deletion requests. Allocating one RES per site allows CCs to make local reconfiguration decisions, which provides more scalable and efficient reconfiguration enforcement.

Finally, Poema can support alternative configuration solutions according to the desired tradeoffs among several requirements, such as scalability, fault tolerance, efficiency, and ease of management. For instance, a centralized directory facilitates policy storage and consistency checking but represents a single point of failure. A centralized RES could handle policy execution on behalf of resource-limited devices, but it might suffer from poor scalability and increase the network traffic due to the coordination between RES and the devices.

Limitations

The current Poema implementation has some limitations. Most significant is the middleware infrastructure's complexity, which might not be appropriate for devices with very limited resources.

The framework described here focuses on devices that can support a standard Java virtual machine.

We have compared the personal assistant application's performance in the Poema-based reconfiguration with the performance in traditional hard-coded reconfiguration directives. Specifically, we evaluated the application's policy response time to adapt to changes—that is, the time needed to complete a reconfiguration action after receipt of the related event.

The Poema-based reconfiguration exhibited a slightly higher response time—about a 10 percent increase—when compared with the hard-coded reconfiguration. The overhead comes from the additional time required to select the triggered policy and parse it for extracting policy applicability conditions and actions.[10] However, Poema's simplification of the runtime application configuration and its reusability counterbalance this difference.

Among emerging current approaches to separate mobility from computational concerns, Poema uniquely supports the specification of different mobility patterns—remote evaluation, code-on-demand, and mobile agent—in a single programming scheme. Application developers can modify reconfiguration choices at both compile/loading time and runtime without affecting the application implementation. Our future work will focus on new techniques for policy conflict detection and resolution and on experiments in other areas of ubiquitous computing, such as managing the resource bindings from software or device component migrations. ■

References

1. A. Fuggetta, G. Picco, and G. Vigna, "Understanding Code Mobility," *IEEE Trans. Software Eng.*, May 1998, pp. 352-361.
2. A. Tripathi et al., "Distributed Collaborations Using Network Mobile Agents," *Proc. 2nd Int'l Symp. Agent Systems and Applications*, LNCS 1882, Springer-Verlag, 2000, pp. 126-137.
3. R.H. Glitho and T. Magedanz, eds., special issue on Applicability of Mobile Agents to Telecommunications, *IEEE Network*, May/June 2002, pp. 6-40.
4. P. Bellavista, A. Corradi, and C. Stefanelli, "Mobile Agent Middleware for Mobile Computing," *Computer*, Mar. 2001, pp. 73-81.
5. S. Wright, R. Chadha, and G. Lapiotis, eds., special issue on Policy-Based Networking, *IEEE Network*, Mar. 2002, pp. 8-56.
6. R. Montanari, G. Tonti, and C. Stefanelli, "A Policy-Based Mobile Agent Infrastructure," *Proc. 3rd IEEE Int'l Symp. Applications and the Internet*, IEEE CS Press, 2003, pp. 370-379.
7. N. Damianou et al., "The Ponder Policy Specification Language," *Proc. 2nd Int'l Workshop Policies for Distributed Systems and Networks*, LNCS 1995, Springer-Verlag, 2001, pp. 18-38.
8. E. Lupu and M. Sloman, "Conflicts in Policy-Based Distributed Systems Management," *IEEE Trans. Software Eng.*, June 1999, pp. 852-869.
9. A.K Bandara, E. Lupu, and A. Russo, "Using Event Calculus to Formalise Policy Specification and Analysis," *Proc. 4th Int'l Workshop Policies for Distributed Systems*, IEEE CS Press, 2003, pp. 26-39.
10. R. Montanari, G. Tonti, and C. Stefanelli,, "Policy-Based Separation of Concerns for Dynamic Code Mobility Management," *Proc. 27th Int'l Conf. Computer Software and Applications*, IEEE CS Press, 2003, pp. 82-90.

Rebecca Montanari *is a research associate of computer engineering at the University of Bologna. Her research focuses on Internet architectures, policy-based systems and service management, mobile agents, security in mobile agent systems, public-key infrastructures, and access-control models. Montanari received a PhD in computer science from the University of Bologna. She is a member of the IEEE and the Italian Association for Computing. Contact her at rmontanari@deis.unibo.it.*

Emil Lupu *is a lecturer in the Department of Computing at Imperial College London, where he leads several projects on policy-based management of distributed systems and networks. His research interests include network and systems management and design, security, and adaptability issues in pervasive systems. Lupu received a PhD in computing from Imperial College London. Contact him at e.c.lupu@imperial.ac.uk.*

Cesare Stefanelli *is an associate professor of computer engineering at the University of Ferrara. His research interests include distributed and mobile computing, mobile code, network and systems management, and network security. Stefanelli received a PhD in computer science from the University of Bologna. He is a member of the IEEE and the Italian Association for Computing. Contact him at cstefanelli@ing.unife.it.*

CALLS FOR PAPERS

ISADS 2005, 7th Int'l Symp. on Autonomous Decentralized Systems, 4-6 Apr., 2005, Chengdu, China. Papers due **15 Aug.** http://isads05.swjtu.edu.cn

ISMVL 2005, 35th Int'l Symp. on Multiple-Valued Logic, 18-21 May, 2005, Calgary, Canada. Papers due **1 Nov.** www.enel.ucalgary.ca/ISMVL2005/cfp.html

CALENDAR

AUGUST 2004

9-10 Aug: MTDT 2004, IEEE Int'l Workshop on Memory Technology, Design, & Testing, San Jose, Calif. www.ece.rochester.edu/workshops/MemoryDesign/

16-19 Aug: CSB 2004, Computational Systems Bioinformatics Conf., Palo Alto, Calif. http://conferences.computer.org/bioinformatics/

19-20 Aug: ISESE 2004, Int'l Symp. on Experimental Software Eng., Redondo Beach, Calif. www.isese.org/

22-24 Aug: Hot Chips 16, Symp. on High-Performance Chips, Palo Alto, Calif. www.hotchips.org/

30 Aug.-1 Sept: NCA 2004,3rd IEEE Int'l Symp. on Network Computing & Applications, Cambridge, Mass. www.ieee-nca.org/

30 Aug.-1 Sept: ICALT 2004, 4th IEEE Int'l Conf. on Advanced Learning Technologies, Joensuu, Finland. http://lttf.ieee.org/icalt2004/

SEPTEMBER 2004

6-9 Sept: 3DPVT 2004, 2nd Int'l Symp. on 3D Data Processing, Visualization, & Transmission, Thessaloniki, Greece. www.umiacs.umd.edu/conferences/3dpvt04/

6-10 Sept: RE 2004, 12th IEEE Int'l Requirements Eng. Conf., Kyoto, Japan. www.re04.org/

7-10 Sept: PARELEC 2004, Int'l Conf. on Parallel Computing in Electrical Eng., Dresden, Germany. www.parelec.org/

8-10 Sept: CODES + ISSS 2004, 2nd IEEE/ACM/IFIP Int'l Conf. on Hardware/Software Codesign & System Synthesis, Stockholm. www.ida.liu.se/conferences/codes/

9-10 Sept: MTV 2004, 5th Int'l Workshop on Microprocessor Test & Verification, Austin, Texas. http://dropzone.tamu.edu/MTV/

11-17 Sept: ICSM 2004, 20th Int'l Conf. on Software Maintenance (with METRICS 2004, SCAM 2004, & WSE 2004), Chicago. www.cs.iit.edu/~icsm2004/

12-14 Sept: BTW 2004, Board Test Workshop, Loveland, Colo. www.molesystems.com/BTW04/

13-15 Sept: CEC04-East, IEEE Conf. on E-Commerce Technology for Dynamic E-Business, Beijing. http://tab.computer.org/tfec/cec04-east/

14-16 Sept: METRICS 2004, 10th Int'l Symp. on Software Metrics, Chicago. www.swmetrics.org/

15-16 Sept: SCAM 2004, 4th IEEE Int'l Workshop on Source Code Analysis & Manipulation, Chicago. www.brunel.ac.uk/~csstmmh2/scam2004/

15-18 Sept: SCC 2004, IEEE Int'l Conf. on Services Computing, Shanghai. http://conferences.computer.org/scc/2004/

20-23 Sept: CLUSTER 2004, IEEE Int'l Conf. on Cluster Computing, San Diego, Calif. http://grail.sdsc.edu/cluster2004/

20-24 Sept: WI-IAT 2004, IEEE/WIC/ACM Int'l Conf. on Web Intelligence & Intelligent Agent Technology, Beijing. www.maebashi-it.org/WI04/

20-24 Sept: EDOC 2004, 8th IEEE Enterprise Distributed Object Computing Conf., Monterey, Calif. www.edocconference.org/

20-25 Sept: ASE 2004, 19th IEEE Int'l Conf. on Automated Software Eng., Linz, Austria. www.ase-conference.org/

26-29 Sept: VL/HCC 2004, IEEE Symp. on Visual Languages & Human-Centric Computing, Rome. http://vlhcc04.dsi.uniroma1.it/

28-30 Sept: SEFM 2004, 2nd IEEE Int'l Conf. on Software Eng. & Formal Methods, Beijing. www.iist.unu.edu/SEFM2004/

29 Sept.-3 Oct: PACT 2004, Int'l Conf. on Parallel Architectures & Compilation Techniques, Antibes Juan-les-Pins, France. www.pactconf.org/

Submission Instructions

The Call and Calendar section lists conferences, symposia, and workshops that the IEEE Computer Society sponsors or cooperates in presenting. Complete instructions for submitting conference or call listings are available at www.computer.org/conferences/submission.htm.

A more complete listing of upcoming computer-related conferences is available at www.computer.org/conferences/.

Code Hacking: A Developer's Guide to Network Security, Richard Conway and Julian Cordingley. The authors detail the software and techniques hackers use and provide a hands-on approach to learning the vital security skills needed to protect network systems. Cutting through cursory issues, the book quickly delves into the essentials at a code and implementation level. It also teaches users how to write and use scanners, sniffers, and exploits, while helping developers write network security test harnesses for applications and infrastructure. In addition to explaining how to create passive defense strategies that collect data on hackers, the authors use techniques such as penetration testing to explain active defense strategies.

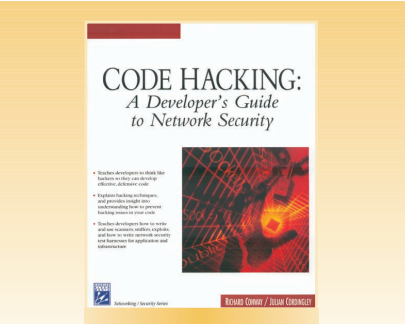
To help developers learn from a variety of perspectives, the book provides software explanations and code samples in C#, C++, Java, and Perl. The companion CD-ROM contains a custom security scanner written in C# that combines a port and vulnerability scanner.

Key features include coverage of essential network security issues including firewalls, authentication, encryption, Web hacking, application hacks, operating system vulnerabilities, writing exploits, and wireless LANs.

Charles River Media; www.charlesriver.com; 1-58450-314-9; 385 pp.; $54.95

System-on-Chip Architectures and Implementations for Private-Key Data Encryption, Máire McLoone and John V. McCanny. Rapid developments in communication systems demand performing data encryption in real time. Acquiring this capability will be key to the successful growth of major applications in areas such as satellite communications and e-commerce.

Methods that implement cryptography in software cannot cope with the



demands of rapidly growing broadband communication systems. Thus, innovative hardware solutions that involve mapping complex mathematical operations onto special-purpose silicon circuit architectures provide the only feasible solution.

This book presents new generic silicon architectures for the DES and Rijndael symmetric key encryption algorithms that developers can use to rapidly and effortlessly generate system-on-chip cores that support numerous application requirements. The authors also describe efficient silicon SHA-1, SHA-2 and HMAC hash algorithm architectures and present a single-chip Internet Protocol Security (IPSec) architecture that combines a generic Rijndael design with a highly efficient HMAC-SHA-1 implementation.

Kluwer Academic/Plenum Publishers; www.wkap.nl/; 0-306-47882-X; 154 pp.; $110.00.

A Classroom of One: How Online Learning Is Changing Our Schools and Colleges, Gene I. Maeroff. This short history of online learning in the US and around the world explores the newly emerging online learning initiatives, from Penn State's World Campus to the Florida Virtual School. Maeroff ultimately provides a snapshot of the way in which technology is changing people's minds with regard to the nature of higher education.

The author looks at methods of electronic delivery, the quality of information being delivered, and the quality of interaction the online delivery engenders. He also explores how learners adapt to this new technology and how much responsibility rests on the student's shoulders. Finally, and maybe tellingly, the author looks at the business aspects of online learning.

Palgrave Macmillan; www.palgrave.com; 1-4039-6537-4; 320 pp.; $16.95.

The Elements of C++ Style, Trevor Misfeldt, Gregory Bumgardner, and Andrew Gray. This book targets all C++ practitioners, with special emphasis on those who work in teams where consistency is critical. Just as Strunk and White's *The Elements of Style* provides rules of usage for writing English, this text furnishes a set of rules for writing C++. The authors offer a collection of standards and guidelines for creating solid C++ code that will be easy to understand, enhance, and maintain.

The book provides conventions for formatting, naming, documentation, programming, and packaging for the latest ANSI standard of C++. It also includes discussion of advanced topics such as templates.

Cambridge University Press; www.cambridge.org; 0-521-89308-9; 190 pp.; $14.00.

Editor: Michael J. Lutz, Rochester Institute of Technology, Rochester, NY; mikelutz@mail.rit.edu. Send press releases and new books to *Computer*, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720; fax +1 714 821 4010; newbooks@computer.org.

Pkware Introduces Secure Zip for Windows

SecureZIP is Pkware's new tool for enterprise environments. It is the fourth generation of the company's Windows-based security utility and the first from the company's new SecureZIP product line. Pkware introduced SecureZIP to give companies a tool for protecting confidential information—with minimal disruption to organizational IT infrastructures. SecureZIP is designed to help IT pros comply with the increasing number of regulations related to data privacy and deal with the mounting pressures to manage and protect proprietary information more effectively.

Contact the company for licensing information; www.pkware.com.

State-of-the-Art Visualization Module

ZyLAB has announced a new visualization tool—the ZyImage Visualization Module—that is designed to help developers view and assess large amounts of structured data. This new visualization module can create dynamic, interactive overviews of structured data. The tooling in the module uses tree overview structures that let users see all the available information on screen at one time; www.zylab.com.

Toolkit Ports Unix Code to 64-Bit Windows

MKS, a provider of tools for porting Unix and Linux applications to Windows, recently announced several 64-bit products, including a beta version of the new 64-bit edition of MKS Toolkit. The beta version offers support for extended-architecture 64-bit platforms such as AMD64 running Microsoft Windows 64-bit operating systems.

The tool is designed to help developers migrate both 32-bit and 64-bit Unix and Linux applications for deployment on 64-bit Windows platforms. MKS Toolkit is available from the MKS site for trial download.

Contact the company for licensing information; www. mks.com.

Please send new product announcements to products@computer.org.

Siemens Offers Real-Time Communication over IP

With the HiPath 8000 Real-Time IP System, Siemens now offers a real-time Web services communication architecture based on the Session Initiation Protocol. SIP is the Internet Engineering Task Force's signaling protocol standard for telephony and conferencing.

The HiPath 800 system uses a hosted, Web-services-communication system that can be deployed and managed from within a data center. It is designed to provide enterprise features with carrier-class reliability and scalability for organizations with hundreds of thousands of users. The system can be deployed and managed from within a data center.

Contact the company for licensing information; www.icn.siemens.com.

American Arium Upgrades SourcePoint Debugging Interface

American Arium, a maker of software debugging tools, announced a new version of SourcePoint, the company's flagship debugging interface. Now in version 7.0, SourcePoint supports Pentium 4 processors at all operating frequencies, Intel 64-bit extensions to 32-bit processors, and AMD Opteron and Althon 64 processors.

The software offers several new features over early versions, including debugging within IDEs and hot-plugging. SourcePoint 7.0 works in concert with the the company's Arium ECM-50 emulator; www.arium.com.

TI Reduces Pricing on DSPs

Texas Instruments announced two new low-cost, high-performance DSPs: the TMS320C6410 and TMS320C6413. Targeting packet-switched telecom—including video-over-IP services and other cost-sensitive, high-performance applications—the new devices are said to provide TMS320C64x performance but with additional memory.

Contact the company for bulk-purchasing information; www.ti.com.

Security Robot Doubles as Wi-Fi Assistant

The new PatrolBot robot from ActivMedia Robotics can integrate with a building's central heating and cooling systems, security systems, air-quality controls, Wi-Fi networks, and even lighting and power systems to provide building services and emergency backup. PatrolBot can also greet guests, guide them to their destinations, or lead building tours.

Once a PatrolBot scans its work areas, it travels automatically to perform tasks: mapping temperatures to improve central heating and cooling efficiency; measuring Wi-Fi signal strength to improve coverage; enabling security guards to investigate problems, and carrying emergency supplies or other equipment into an unsafe or dark building.

PatrolBots are available directly from ActivMedia Robotics and cost $30,000; www.MobileRobots.com.

The PatrolBot security robot uses ActivMedia Robotics' laser-mapping system to navigate building corridors. The system maps buildings on the fly and lets the robot move to dynamically controlled building destinations without first requiring retrofitting.

Top 10 Student Teams Vie for $15,000 CSIDC Prize

The Fifth Annual Computer Society International Design Competition features the work of teams from all areas of the world, including Brazil, Nepal, South Africa, and the US.

A computer design event that has spanned two semesters and involves undergraduates from around the world is at the live demo stage. The IEEE Computer Society International Design Competition allows teams of undergraduate engineering students the opportunity to design, from inception to prototype, a special-purpose computer-based device to solve a real-world problem. The competition this year focused on the theme of "Making the World a Safer Place."

An international panel of judges selected ten teams from more than 250 original contenders to face off for two intensive days of competition. In May, judges reviewed 80 project reports containing project specifications, including engineering considerations and implementation. Many teams failed to pass an interim report stage in February, were eliminated through intra-school competitions, or dropped out before submitting the 20-page report. (CSIDC rules allow only one team per school to advance to the final stages of the competition.)

Prototype student-developed devices on display at the World Finals include a child abduction prevention system, several earthquake warning systems, and a pollution reduction device. The "Top 10 CSIDC Teams" sidebar lists the CSIDC 2004 World Finals teams and project names.

Said Alan Clements, chair of the CSDIC Committee for four years and a professor at the University of Teesside in England, "Some of the best of the teams in 2004 have applied computer technology to ... areas such as earthquake detection and water pollution. Their ability to investigate problems of such importance to the community is truly impressive. Indeed, we have decided to follow their lead and make CSIDC 2005 a multidisciplinary competition that requires all teams to take such an approach."

CSIDC prizes range from $15,000 for first place to $6,000 for third place, plus honorable mention awards of $2,000. The competition also includes two special prizes: the Microsoft Award for Software Engineering and the Microsoft Multimedia Award.

Primary financial support for CSIDC 2004 is provided by Microsoft, which has provided $1,000,000 in funding for CSIDC until 2006. ABB and the IEEE Foundation have provided additional financial support. ■

Top 10 CSIDC Teams

The IEEE Computer Society International Design Competition encourages undergraduate teams to work together much as they would at industry engineering posts. Addressing the broad theme of "Making the World a Safer Place," these 10 teams from around the world created prototype systems that impressed an international panel of judges enough to earn them a spot in the CSIDC World Finals live event.

- Humboldt University, Berlin (Germany), "PLAS.MA: Person Loss Avoidance System for Mobile Applications"
- Institute of Engineering, Pulchowk Campus, Tribhuvan University (Nepal), "TremorFlash: Earthquake Warning and Follow-up System"
- Instituto Militar de Engenharia (Brazil), "Dangerous Load Monitoring Alarm System"
- Iowa State University (USA), "Spatial Cue"
- Lahore University of Management Sciences (Pakistan), "SensUS Structure Security System"
- National Taiwan University (Taiwan), "ADaptable VIsionary System for Earthquake Resolution"
- Politehnica University of Bucharest (Romania), "eXpress! Help"
- Poznan University of Technology (Poland), "Lifetch"
- University of Pretoria (South Africa), "Intelligent Elderly Care—iEC"
- University of Virginia (USA), "The Polluter Must Pay"

Computer Society Awards Program Honors Achievements in Many Computing Fields

With award recognitions large and small, the IEEE Computer Society demonstrates appreciation for the contributions that our members and others have made to the computing field. This month, the work of researchers in three discrete areas takes center stage. Jack Dongarra was a primary contributor to Linpack and other open source software packages. Victor Basili is active in the software quality assurance community. And Carlo Séquin has made a name for himself in the computer graphics field.

Jack J. Dongarra Receives Fernbach Award

Jack J. Dongarra introduced the Linback benchmark for high-performance computers.

In recognition of his work at the nexus of mathematics and computing, Jack J. Dongarra has received the Computer Society's Sidney Fernbach Memorial Award. Dongarra's research focuses on numerical algorithms in linear algebra and the use of advanced computer architectures, among other fields of study. A former senior scientist at Argonne National Laboratory, Dongarra now holds appointments as University Distinguished Professor of Computer Science at the University of Tennessee, Distinguished Research Staff member at Oak Ridge National Laboratory, and Adjunct Professor at Rice University. He is the founder and director of the Innovative Computing Laboratory at the University of Tennessee's Knoxville campus. The ICL uses high-performance computing tools to tackle science's most challenging problems.

A key contributor to the design and implementation of the Linpack collection of Fortran subroutines that solve linear least-squares problems, Dongarra has also participated in the development of similar open-source packages including Lapack, Netlib, and Atlas. Current applications of high-performance computing techniques that Dongarra is investigating include climate modeling and materials science.

Dongarra is a Fellow of the IEEE, the AAAS, and the ACM and is a member of the National Academy of Engineering. His award citation reads, "For outstanding and sustained contributions to the area of mathematical software, most particularly in the areas of communication and numerical libraries and performance benchmarks for high-performance computing."

Established in 1992, the Fernbach Award recognizes contributions in the application of high-performance computers. Awardees receive a certificate and a $2,000 honorarium.

Maryland's Victor R. Basili Honored with Mills Award

Victor R. Basili is a cofounder of the International Software Engineering Research Network.

Victor Basili, a professor at the University of Maryland's Institute for Advanced Computer Studies, has re-

Fall 2004 CSDP Testing Window Opens 1 September

Attention software professionals: Do you need an extra edge to prove your skills to current or potential employers? Apply to sit for the IEEE Computer Society Certified Software Development Professional (CSDP) credential. Each year, the Computer Society offers two opportunities to take the CSDP exam: April through June, and September through October.

Software engineers who hold a baccalaureate degree, and who have a minimum of 9,000 hours of experience in the field, are eligible to apply. In addition, candidates for certification must have had at least two years of software engineering experience within the four-year period prior to application.

Thomson Prometric administers the CSDP exam at test centers throughout the world. For IEEE or Computer Society members, 2004 CSDP exam fees total $400, including a $100 application fee and a $300 test administration fee. A CSDP Study Group online Yahoo forum, linked from the CSDP Web site, can help you prepare. Resource materials and an online test preparation class are also available. Register for the prep class by **1 August** to receive a $100 discount on course fees.

Applications for the 1 September through 30 October testing window must be postmarked by **15 August**. The application form is available online at www.computer.org/certification/bulletin.htm. For general information on the IEEE Computer Society CSDP program, visit www.computer.org/certification/.

ceived the IEEE Computer Society Harlan D. Mills Award for his contributions to software engineering.

Basili serves as director of the NASA Goddard Space Flight Center's Software Engineering Laboratory (SEL) and is the executive director of the Fraunhofer Center for Experimental Software Engineering–Maryland.

A founding member of the International Software Engineering Research Network (ISERN), Basili's research interests include experimental software engineering and software quality assurance. He has written several books—including one that he coauthored with Harlan Mills—on structured programming, the SIMPL-T programming language, and related topics.

In 1994, Basili's SEL team won the IEEE Computer Society Award for Software Process Achievement. An IEEE Computer Society Golden Core member and active volunteer, Basili is also a Fellow of the IEEE and the ACM. His citation for the Mills award reads, "For significant contributions to programming languages, program reading and writing, and empirical methods."

Established in 1999, the Mills award recognizes researchers and practitioners who have demonstrated long-standing, sustained, and meaningful contributions to the theory and practice of the information sciences. Recipients are presented with a memento and a $3,000 honorarium.

Carlo H. Séquin of Berkeley Wins Technical Achievement Award

Carlo H. Séquin applies CAD/CAM technology to the creation of abstract sculpture.

The IEEE Computer Society has honored computer graphics researcher Carlo Séquin, professor and associate dean at the University of California, Berkeley, with a Technical Achievement Award for his work in computer-aided design.

Coming from an experimental physics background, Séquin got his first taste for computer graphics work while at Bell Labs nearly 30 years ago. More recently, Séquin developed the Scherk-Collins Sculpture Generator, a computer program that helps sculptors visualize different configurations on a virtual sculpture so that they can determine the specifications necessary before creating physical pieces of art.

In cooperation with Berkeley mechanical engineering professor Paul Wright, under a grant from the US National Science Foundation, Séquin is currently investigating methods for using advanced CAD/CAM tools to reduce the number of steps between the conceptualization and design stage of a product and its commercial manufacture.

Séquin's Technical Achievement Award citation reads, "For outstanding contributions to design tools for computer architectures, computer graphics, industrial CAD/CAM, and artistic expressionism." A Fellow of both the IEEE and the ACM, Séquin is also an elected member of the Swiss Academy of Engineering Sciences.

The Computer Society Technical Achievement Award is presented for outstanding and innovative recent contributions in computer engineering.

From algorithms to artist's tools, the IEEE Computer Society Awards program recognizes achievements in all areas of computing. To nominate a colleague for his or her unique contributions to any computing specialty, visit www.computer.org/awards/. Most nomination forms are due by **1 October.**

Call for Petition Candidates

The IEEE Computer Society welcomes the nominations of candidates for office. To have a name added to the ballot, a member can submit a petition to the Society secretary via mail, fax, or e-mail indicating the desired office, the starting date of the term, and the name of the candidate. The petition must also include the signatures of voting members of the Society: at least 250 signatures for Board term nominees and at least 1,000 signatures for officer nominees. Petition "signatures" can simply indicate the signing member's name and member number. A voting member can sign only one Board of Governors petition and only one officer petition for each other office. All petitions were to have been submitted to the Society secretary by the 2 June deadline.

For each petition nomination, the Society secretary must receive a statement signed by the nominee indicating a willingness and availability to serve if elected. Petition candidates must also submit biographical data, position statements, and 300-dpi digital images or studio-quality head-and-shoulders photographs to the Society secretary.

All petition nominee materials must be received by **31 July**. Send them to Computer Society Secretary Oscar Garcia at the IEEE Computer Society Headquarters, 1730 Massachusetts Ave. NW, Washington DC 20046-1992; o.garcia@computer.org.

Editor: Bob Ward, *Computer*, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; bnward@computer.org

IEEE Computer Society Election Update

Beginning next month, all members of the IEEE Computer Society will have the opportunity to vote for the officers who will plan new activities and direct the Society's operations in 2005. Members elected to volunteer posts will serve terms on the Board of Governors and as leaders of Boards that oversee projects such as our publications activities, educational programs, and electronic media resources. These volunteers will serve under 2005 President Gerald L. Engel, chosen as president-elect in last year's campaign.

Society members can become candidates for office in one of two ways: by Nominations Committee recommendation or by petition. The Nominations Committee accepted member recommendations of candidates until April. At a June meeting, the current Board of Governors approved the following slate of candidates brought forward by the Nominations Committee. For rules on becoming a petition candidate, see the "Call for Petition Candidates" sidebar.

PRESIDENT-ELECT AND VICE PRESIDENT CANDIDATES

The Board-approved candidates for 2005 president-elect/2006 president are Deborah M. Cooper and Yervant Zorian. The president supervises decisions that affect the Society's programs and operations and is a nonvoting member on most Society program boards and committees.

Candidates for first vice president are Rangachar Kasturi and Michael R. Williams. The second vice president candidates are James W. Moore and Murali R. Varanasi. After the elections, 2005 President Gerald Engel will appoint the two elected vice presidents to oversee two Society boards. At his discretion, Engel will select his own appointees to head up the Society's other governing boards.

BOARD OF GOVERNORS CANDIDATES

Members of the Board of Governors serve rotating three-year terms. Candidates for Board of Governors terms from 2005 to 2007 are Jean M. Bacon, Donald J. Bagert, George V. Cybenko, Reiner W. Hartenstein, James D. Isaak, Richard A. Kemmerer, Susan K. Land, Itaru Mimura, Brian M. O'Connell, Sorel R. Reisman, Christina M. Schober, Robert H. Sloan, and Ronald J. Vetter. The seven candidates who receive the most votes will assume positions on the Board starting in January 2005.

The IEEE Computer Society election window begins on 13 August and ends on 4 October. All members will have the opportunity to vote via paper mail or online balloting. We encourage all members to take part in electing the leaders of your Computer Society. Visit www.computer.org/election/ after 13 August to cast your vote. ■

THE UNIVERSITY OF TENNESSEE, The Imaging, Robotics, and Intelligent Systems (IRIS) Laboratory. The IRIS Lab invites applicants for multi-year Research Assistant/Associate Professorships and Ph.D. Fellowships. The IRIS Lab's emphasis is in the fields of Three-dimensional Imaging, Data Fusion, and Visualization. For 2004, the IRIS Lab is expected to have a staff of 50 and an annual budget over $3.5Million. Interested persons should contact: Mongi Abidi, Professor and Associate Department Head, Department of Electrical and Computer Engineering, 328 Ferris Hall, Knoxville, TN 37996-2100. Web: http://imaging.utk.edu/opportunities/opportunities.htm, Phone: 865-974-5454, Fax: 865-974-5459, E-Mail: abidi@utk.edu. UTK is an EE/AA/Title VI/Title IX/Section 504/ADA/ADEA Employer.

THE UNIVERSITY OF TOLEDO, Computer Science and Engineering Technology. The Department of Engineering Technology in the College of Engineering at The University of Toledo seeks candidates with a background in programming, computer architecture, computer operating systems, and computer networking for a tenure-track faculty position in Computer Science and Engineering Technology (CSET). Candidates for this position must have strong commitment to undergraduate education. A PhD in computer science is required for this appointment. A minimum of three years of industry experience is preferred. Founded in 1872, The University of Toledo (www.utoledo.edu) is a large, multi-faceted metropolitan research institution. UT provides high quality academic opportunities for its students, and emphasizes engagement in the economic, social and cultural development of its region and focused growth in research. The College of Engineering (www.eng.utoledo.edu) currently has 88 faculty members with a combined enrollment of approximately 3000 undergraduate and graduate students. The Engineering Technology Department (www.eng.utoledo.edu/eng-tech) has 20 faculty members in four degree programs with an enrollment of 1100 students, while the CSET program has 5 faculty members with an enrollment of 325 students. Our state of the art laboratories, renowned faculty, industry collaborations, and multi-disciplinary programs offer an exceptional opportunity for those joining our team. Please send curriculum vitae, research and teaching interests, and the names of three references to CSET Faculty Search Committee, Attn: Mrs. Patricia Mowery, Office of the Dean, College of Engineering, MS310, The Univer-

D. E. Shaw Research and Development

Systems Architects and ASIC Engineers

Specialized Supercomputer for Computational Drug Design

Extraordinarily gifted systems architects and ASIC design and verification engineers are sought to participate in the development of a special-purpose supercomputer designed to fundamentally transform the process of drug discovery within the pharmaceutical industry. This early-stage, rapidly growing project is being financed by the D. E. Shaw group, an investment and technology development firm with approximately US $8 billion in aggregate capital. The project was initiated by the firm's founder, Dr. David E. Shaw, and operates under his direct scientific leadership.

This project aims to combine an innovative, massively parallel architecture incorporating 90-nanometer "system on a chip" ASICs with novel mathematical techniques and groundbreaking algorithmic advances in computational biochemistry to direct unprecedented computational power toward the solution of key scientific and technical problems in the field of molecular design. Successful candidates will be working closely with a number of the world's leading computational chemists and biologists, and will have the opportunity not only to participate in an exciting entrepreneurial venture with considerable economic potential, but to make fundamental contributions within the fields of biology, chemistry, and medicine.

The candidates we seek will be unusually intelligent and accomplished, with a demonstrated ability to design and implement complex, high-performance hardware solutions based on the latest semi-custom technologies. We are prepared to reward exceptionally well-qualified individuals with above-market compensation.

Please send resume, along with GPAs, standardized test scores (SAT, GRE), and compensation history, to career9@desrad.deshaw.com.

D. E. Shaw Research and Development, L.L.C. does not discriminate in employment matters on the basis of race, color, religion, gender, national origin, age, military service eligibility, veteran status, sexual orientation, marital status, disability, or any other protected class.

D E Shaw & Co

NTT DoCoMo

DoCoMo USA Labs

World Class Opportunities at DoCoMo USA Labs

DoCoMo Communications Laboratories USA, Inc.

DoCoMo USA Labs is part of the world's leading mobile communications company, NTT DoCoMo, the premier wireless service provider in Japan and an innovator recognized throughout the world of mobile communications. NTT DoCoMo launched the world's first 3G mobile service in 2001 and operates the world's largest mobile internet service, i-mode. The company conducts its Research and Development at Yokosuka Research Park in Tokyo and at three global R&D locations, San Jose USA, Munich Germany and Beijing China.

DoCoMo USA Labs is located in Silicon Valley where we are conducting research into advanced operating systems, wireless applications, networks, protocols and media. The laboratory is organized into four distinct labs: the Media Lab (ML), the Mobile Software Lab (MSL), the Network Architecture Lab (NAL), and the Network Service and Security Lab (NSSL). The four labs are unified by the common goal of providing breakthroughs in mobile Internet technologies and future platform definition.

This is your opportunity to be part of the future and pursue your dreams at DoCoMo USA Labs. We are expanding and seeking suitably qualified applicants for the following positions.

Network Architecture Lab (nal@docomolabs-usa.com)

- **Wireless Networking Project Manager (ref. NAL1s)**
  Address fundamental research in wireless networks, protocols and systems. IP-based RAN and core network design including routing and addressing. MAC layer and mobility management and security.

Mobile Software Lab (msl@docomolabs-usa.com)

- **Mobile Operating Systems Project Manager (ref. MSL1s)**
  Lead research into new operating systems principles and concepts. Topics include memory management, scheduling, file systems, and communications.
- **Secure Languages Researcher (ref. MSL3s)**
- **Operating System Researcher in Kernel Design and Implementation (ref. MSL4s)**
- **Middleware Researcher (ref. MSL6s)**

Network Service and Security Lab (nss@docomolabs-usa.com)

- **Mobile Network Security Senior Researcher (ref. NSSL3s)**
- **Mobile Network Security Researcher (ref. NSSL4s)**
- **Mobile Middleware & Web Services (ref. NSSL2s)**

Media Lab (mm@docomolabs-usa.com)

- **Mobile Multimedia Researcher (ref. MM1s)**
  Speech and audio coding, video coding, non-linear signal processing and delivery of multimedia content over IP networks.

Qualifications and Experience

Applicants are expected to have a Ph.D. or equivalent in C. S., E.E., or C.E. Applications are invited from candidates with all levels of experience. A track record of research is expected for senior positions.

To Apply

Please apply by emailing your resume to the email address shown after each Lab's name and be sure to include job title and job reference number. Example: if you are applying for Mobile Operating Systems Project Manager (ref. MSL1s) you would email to msl@docomolabs-usa.com and put Project Manager / ref. MSL1s in the subject title of the email.

For further information, visit the careers section on our website

www.docomolabs-usa.com

An Equal Opportunity Employer

sity of Toledo, Toledo, OH 43606-3390. Fax: (419)530-8006. Email: pmowery@eng.utoledo.edu. The position will remain open, and applications will be reviewed as received or until appointments are made. The University of Toledo is an affirmative action/equal opportunity employer. Women and minorities are encouraged to apply.

THE UNIVERSITY OF GRONINGEN, department of Mathematics and Computing Science, is currently recruiting an assistant professor in Software Engineering (tenure-track). The Faculty of Mathematics and Natural Sciences is offering young, talented researchers positions which are on the same level as those of assistant professor via the tenure-track system. Researchers are given the opportunity to develop their own line of research within a particular field. The faculty's career policy is characterized by flexible personnel management with a focus on the individual. Academic achievements are seen as being central to the academic career, and ample opportunities for professional development and supplementary training and education are offered. Arrangements for training in the area of teaching will be made with all new employees. Contact prof. dr. ir. J. Bosch, tel. +31 50 3633941, email: j.bosch@cs.rug.nl. http://www.rug.nl/wiskunde/vacatures/vacaturesIWI

CALTECH. Postdoctoral research positions at Caltech's Center for Advanced Computing Research are open in innovative, high performance computer architecture. Ph.D. in computer science, computer engineering or equivalent required. See (www.cacr.caltech.edu/employment/aca-postdoc.html). Resume to: Susan Powell, spowell@cacr.caltech.edu

COMPUTER SCIENCE CHAIR

The **B. THOMAS GOLISANO COLLEGE OF COMPUTING AND INFORMATION SCIENCES (GCCIS)** at **ROCHESTER INSTITUTE OF TECHNOLOGY** is pleased to invite applications for the position of Chair of the Computer Science department. The anticipated start date for this position will be no later than January 1, 2005. The successful candidate is expected to exhibit academic leadership, must be an accomplished and enthusiastic teacher and scholar, have a *broad knowledge of computing and its role in the information age*, and have the ability to contribute in meaningful ways to the Institute's commitment to cultural diversity and pluralism. Candidates must have the credentials, experience, and achievements appropriate to be appointed at the Full Professor level, including a Ph.D. in computer science or closely related area. A recognized research record is essential.

GCCIS is RIT's newest college at the 1,300-acre suburban university located in the beautiful Finger Lakes region in Rochester, New York. In addition to CS, the college is home to the Information Technology and Software Engineering departments, as well as the Laboratory for Applied Computing, the research arm of the college. The college, with 90+ faculty members, is housed in a new 126,500 square foot state-of-the art building. Faculty in the Computer Science department are engaged in scholarly activities in such areas as data mining and discovery informatics, intelligent systems, complexity theory and cryptography, graphics, distributed systems, among others. Detailed information can be found at http://www.cs.rit.edu. The department is expected to play a central role in the development of the new college-level PhD program.

Review of applications will begin on July 1, 2004. Candidates are strongly encouraged to submit their applications electronically. The position will remain open until filled. Applications must include a summary of education and professional background, a list of publications, a summary of teaching and research experience, the names of three references, and a brief essay on the role of computer science within computing and strategic future directions.

Dean's Office – CS Chair Search
B. Thomas Golisano College of
Computing and Information Sciences
Rochester Institute of Technology
20 Lomb Memorial Drive
Rochester, NY 14623
http://www.rit.edu/~gccis
Email: cssearch@gccis.rit.edu

R·I·T

"providing career education over a lifetime"
RIT is an Affirmative Action/Equal
Employment Opportunity Employer.

SUBMISSION DETAILS: Rates are $275.00 per column inch ($300 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.

ADVERTISER / PRODUCT INDEX JULY 2004

Advertiser	Page Number
ABB Switzerland	88
D.E. Shaw	89
DoCoMo USA Labs	89
Embedded Systems 2004	5
IEEE	Cover 3
IPDPS 2005	Cover 2
Prediction Company	91
RIT	90
Seapine Software, Inc.	Cover 4

Advertising Personnel

Marion Delaney
IEEE Media, Advertising Director
Phone: +1 212 419 7766
Fax: +1 212 419 7589
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

Advertising Sales Representatives

Mid Atlantic (product/recruitment)
Dawn Becker
Phone: +1 732 772 0160
Fax: +1 732 772 0161
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)
Robert Zwick
Phone: +1 212 419 7765
Fax: +1 212 419 7570
Email: r.zwick@ieee.org

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

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

Southeast (recruitment)
Jana Smith
Phone: +1 404 256 3800
Fax: +1 404 255 7942
Email: jsmith@bmmatlanta.com

Southeast (product)
Bob Doran
Phone: +1 770 587 9421
Fax: +1 770 587 9501
Email: bd.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)
Josh Mayer
Phone: +1 972 423 5507
Fax: +1 972 423 6858
Email: josh.mayer@wageneckassociates.com

Northwest (product)
Peter D. Scott
Phone: +1 415 421-7950
Fax: +1 415 398-4156
Email: peterd@pscottassoc.com

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

Japan
Sandy Brown
Phone: +1 714 821 8380
Fax: +1 714 821 4010
Email: sbrown@computer.org

Europe (product)
Hilary Turnbull
Phone: +44 1875 825700
Fax: +44 1875 825701
Email: impress@impressmedia.com

Europe (recruitment)
Penny Lee
Phone: +20 7405 7577
Fax: +20 7405 7506
Email: reception@essentialmedia.co.uk

A Critical Look at Open Source

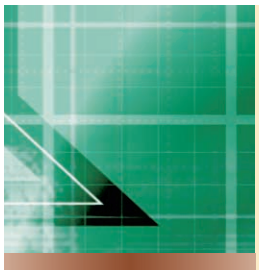
Brian Fitzgerald, University of Limerick

Despite 50 years of research and practice, the development of software—particularly complex and innovative software—is far from a predictable, exact process. Many software projects exceed their budget, fail to conform to their development schedule, and do not meet expectations when eventually delivered.

In recent years, open source software—more properly, free and open source software—has emerged as one popular solution to the so-called "software crisis." Advocates regard F/OSS as an agile, practice-led initiative that addresses three key issues:

- *Cost*: F/OSS products are usually freely available for public download.
- *Time scale*: The collaborative, parallel efforts of globally distributed developers allow many F/OSS products to be developed much more quickly than conventional software.
- *Quality*: Many F/OSS offerings are recognized for their high standards of reliability, efficiency, and robustness; products such as GNU/Linux, Apache, and Bind have become "category killers" stifling the incentive to develop any competing products.

Supporters also point out that F/OSS harnesses the scarcest resource of all: talented software developers, many of whom exhibit a long-standing commitment to their chosen projects. In addition, the resulting peer review process helps ensure the quality of the software produced.



Despite many challenges, free and open source software remains a viable model.

Moreover, researchers in other fields—including sociology, economics, management, psychology, public policy, and the law—have embraced F/OSS as an innovative model for organizing activity in a variety of social contexts.

F/OSS CHALLENGES

Despite its wide appeal, F/OSS faces a number of serious challenges that have led some naysayers within the software engineering community to declare that the model is doomed to ultimately fail.

Development talent

The main F/OSS contributors are "code gods" acknowledged to be among the world's most talented and highly motivated programmers. The software industry has long recognized the potential contribution of gifted individuals, but they are critical to F/OSS: The absence of face-to-face interaction or other normal organizational sanctions makes it vital that there be an ultimate arbiter with unquestioned authority and ability who charismatically inspires others and can resolve disputes and prevent forking.

However, the widespread interest in F/OSS may exceed the development talent available. To some extent, this is already happening. For example, SourceXchange, a brokering service for companies soliciting F/OSS developers, ceased operations in 2001. Studies of Freshmeat.net and SourceForge.net, two popular F/OSS development Web sites, revealed that most projects have only one or two developers. They also have little apparent vitality, as follow-up studies reported no change in version number or size of code base for many listed projects several months later.

Code quality

While F/OSS pioneers may have been "best-of-breed" programmers, the ability of future contributors is much more in doubt. The popularity of F/OSS increases the risk that so-called net-negative-producing programmers will become involved. Despite the long probationary period programmers serve on many F/OSS projects, NNPs could introduce errors into the code base. This could irremediably harm F/OSS projects, which lack the marketing muscle to help undo damage to volunteers' reputations. Some recent empirical studies question the axiom that all F/OSS products are of high quality.

Project initiation

Even if more companies decide to make their source code generally available, F/OSS developers are not privy to

the organizational design decisions underlying the code base's evolution. Simply releasing a large proprietary product's source code—as Netscape did with its Mozilla browser—is not alone enough to launch a vibrant, fully functioning F/OSS project.

Code modularity

Code modularity is a key to F/OSS success as it is the basis for distributing the development process. However, it can also be a project's Achilles' heel. Excessive modularity increases the risk of *common coupling*, an insidious problem in which modules unnecessarily refer to variables and structures in other modules. Thus, changes to data structures and variables in seemingly unrelated modules can have major knock-on implications. In this way, as F/OSS systems evolve, maintaining them can become difficult if not impossible in the long run.

Mundane tasks

Many software tasks—such as documentation, testing, internationalization/localization, and field support—are tedious but vital, particularly as new cohorts of developers join and maintain projects. However, F/OSS developers could cherry-pick more exciting tasks such as code design, which is understandable in a culture that does not worship the likes of "documentation gods." Unfortunately, nontechnical F/OSS contributors and users may not fill the gap to the extent originally predicted.

Stability

Companies have valid concerns about the survival of and continued support for F/OSS products. The traditional telephone hotline and maintenance contract offer a comfort factor that a voluntary bulletin board—which is the main support for many F/OSS products—cannot provide.

Standardization

GNU/Linux version proliferation already poses a substantial challenge to software providers, who must write and test applications for the many commercial versions available. Also, the independent development of F/OSS products results in time-consuming interoperability and compatibility problems among different product versions.

Standardization is arguably more important for F/OSS developers, who typically do not meet face to face, than for traditional proprietary developers. Any mechanism that can facilitate collective action, such as common standards for integration, would benefit the F/OSS community. Although numerous initiatives are moving broadly in this direction, their conflicting political agendas do not make the prospects of any near-term convergence on standards likely.

WHY F/OSS?

Despite these potential choke points, the future for F/OSS remains bright. Recent research effectively dispels the stereotype of F/OSS developers as anarchistic hackers operating on the fringes of society. Studies by Karim Lakhani and Bob Wolf in the United States and by Rishab Aiyer Ghosh in Europe indicate that most developers are in stable cohabiting relationships, often with children, and are experienced IT professionals paid for their work on F/OSS projects.

Individuals participate in and support F/OSS development for a number of reasons. Developers can collaborate with peers they truly respect, acquiring more rapid feedback and direct recognition through credit acknowledgments on the project lists. In addition, the informal development style of F/OSS liberates many from the formalized organizational environments in which they work every day. Also, students can boost their career prospects by gaining experience on real development projects.

Finally, research suggests that implementing F/OSS can result in significant savings. Organizations that deploy F/OSS products freely offer advice to one another, sharing insights and lessons learned. At the same time, hands-on users must proactively work with IT staff to ascertain whether available F/OSS products meet their needs. This cooperative spirit, which in part may arise from a sense of shared adventure, starkly contrasts with the tradi-

Further Reading

The following resources provide more information on the issues surrounding free and open source software:

- A. Capiluppi, P. Lago, and M. Morisio, "Evidences in the Evolution of OS Projects through Changelog Analyses," *Taking Stock of the Bazaar: 3rd Workshop Open Source Software Eng.*, 25th Int'l Conf. Software Eng., 2003, pp. 19-24; http://opensource.ucc.ie/icse2003.
- B. Fitzgerald and T. Kenny, "Developing an Information Systems Infrastructure with Open Source Software," *IEEE Software*, vol. 21, no. 1, 2004, pp. 50-55.
- E.S. Raymond, *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly & Assoc., 1999.
- S. Rusovan, M. Lawford, and D. Parnas, "Open Source Software Development: Future or Fad?" to be published in J. Feller et al., eds., *Perspectives on Free and Open Source Software*, MIT Press, 2005.
- S.R. Schach and A.J. Offutt, "On the Nonmaintainability of Open-Source Software," *Meeting Challenges and Surviving Success: 2nd Workshop Open Source Software Engineering*, 24th Int'l Conf. Software Eng., 2002; http://opensource.ucc.ie/icse2002/SchachOffutt.pdf.

tional impersonal, bilateral relationship between vendor and customer.

Some argue that F/OSS represents a paradigm shift in software engineering. For example, Eric Raymond describes the disciplined and coordinated conventional approach as a "cathedral" and F/OSS as a noisy, confused "bazaar."

At first glance, F/OSS appears to contravene fundamental tenets of software engineering: no formal requirements specification or design process, no risk assessment or measurable goals, informal coordination and control, and much duplication and parallel effort. However, well-established software engineering principles lie at the heart of F/OSS.

F/OSS does not represent a paradigm shift in software engineering.

Code modularity allows partitioning work among a global pool of developers and facilitates the recruitment of new contributors, as it reduces their learning curve to a subset of modules rather than the entire project. Indeed, developers rewrote projects such as Sendmail, Samba, and even Linux in modular form to ensure successful development.

F/OSS developers also carefully vet and incorporate contributions in accordance with sound configuration management, independent peer review, and testing—all familiar software engineering concepts. ■

Brian Fitzgerald *holds the Frederick A. Krehbiel II Chair in Innovation in Global Business and Technology at the University of Limerick, Ireland. Contact him at bf@ul.ie.*

Editor: Richard G. Mathieu, Dept. of Decision Sciences and MIS, St. Louis University, St. Louis, MO 63108; mathieur@slu.edu

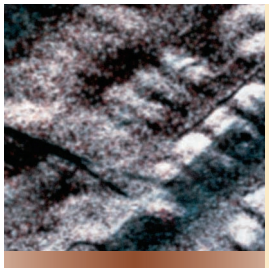
Embedded System Security

Philip Koopman, Carnegie Mellon University

From cars to cell phones, video equipment to MP3 players, and dishwashers to home thermostats—embedded computers increasingly permeate our lives. But security for these systems is an open question and could prove a more difficult long-term problem than security does today for desktop and enterprise computing.

Security issues are nothing new for embedded systems. In 2001, Peter Shipley and Simson L. Garfinkel reported finding an unprotected modem line to a system that controlled a high-voltage power transmission line ("An Analysis of Dial-Up Modems and Vulnerabilities," 2001; www.dis.org/filez/Wardial_ShipleyGarfinkel.pdf). However, as more embedded systems are connected to the Internet, the potential damages from such vulnerabilities scale up dramatically.

This issue is already upon us. Today you can buy Internet-enabled home appliances and security systems, and some hospitals use wireless IP networks for patient care equipment. Cars will inevitably have indirect Internet connections—via a firewall or two—to safety-critical control systems. There have already been proposals for using wireless roadside transmitters to send real-time speed limit changes to engine control computers. There is even a proposal for passenger jets to use IP for their primary flight controls, just a few firewalls away from passengers cruising the Web (Aeronautical Radio, Inc., "Draft 1 of Project Paper 664, 'Aircraft Data Networks,' Part 5, 'Network Interconnection Devices,'" AEEC Letter 01-112/SAI/742, 2 May 2001).



Security for embedded systems involves issues beyond those problems currently being addressed for enterprise and desktop computing.

WHAT'S DIFFERENT ABOUT EMBEDDED SECURITY?

Internet connections expose applications to intrusions and malicious attacks. Unfortunately, security techniques developed for enterprise and desktop computing might not satisfy embedded application requirements.

Cost sensitivity

Embedded systems are often highly cost sensitive—even five cents can make a big difference when building millions of units per year. For this reason, most CPUs manufactured worldwide use 4- and 8-bit processors, which have limited room for security overhead. Many 8-bit microcontrollers, for example, can't store a big cryptographic key. This can make best practices from the enterprise world too expensive to be practical in embedded applications.

Cutting corners on security to reduce hardware costs can give a competitor a market advantage for price-sensitive products. And if there is no quantitative measure of security before a product is deployed, who is to say how much to spend on it?

Interactive matters

Many embedded systems interact with the real world. A security breach thus can result in physical side effects, including property damage, personal injury, and even death. Backing out financial transactions can repair some enterprise security breaches, but reversing a car crash isn't possible.

Unlike transaction-oriented enterprise computing, embedded systems often perform periodic computations to run control loops with real-time deadlines. Speeds can easily reach 20 loops per second even for mundane tasks. When a delay of only a fraction of a second can cause a loss of control-loop stability, systems become vulnerable to attacks designed to disrupt system timing.

Embedded systems often have no real system administrator. Who's the sysadmin for an Internet-connected washing machine? Who will ensure that only strong passwords are used? How is a security update handled? What if an attacker takes over the washing machine and uses it as a platform to launch distributed denial-of-service (DoS) attacks against a government agency?

Energy constraints

Embedded systems often have significant energy constraints, and many are battery powered. Some embedded systems can get a fresh battery charge daily, but others must last months or years on a single battery.

By seeking to drain the battery, an attacker can cause system failure even when breaking into the system is impossible. This vulnerability is critical, for example, in battery-powered devices that use power-hungry wireless communication.

Development environment

Many embedded systems are created by small development teams or even lone engineers. Organizations that write only a few kilobytes of code per year usually can't afford a security specialist and often don't realize they need one.

However, even seemingly trivial programs may need to provide some level of security assurance. Until standard development practice includes rigorous security analysis, developers may overlook even the solutions already available.

EXAMPLE: INTERNET THERMOSTATS

Because embedded systems can effect changes in the physical world, the consequences of exploiting their security vulnerabilities can go beyond mere annoyance to significant societal disruption.

Let's dispense with the most obvious potential attack first. Attackers that break into a computer and get complete control of it can do anything they want with the attached sensors and actuators—send commands to traffic lights, shut down power stations, and so on.

We could argue that industrial-strength security approaches will take care of the really big systems, but smaller systems are less likely to receive lavish attention.

Consider, for example, the household thermostat, which controls heating and cooling. Many have an embedded computer that adjusts the set point a few times each day to keep the house comfortable when people are present and to save energy when they aren't.

Some thermostats let a homeowner use the Internet, perhaps via cell phone, to communicate imminent arrival home after a vacation or a day at work. This gives the thermostat time to reach a comfortable temperature before the owner actually arrives.

However, allowing Internet control of a thermostat gives rise to several potential attacks.

Centralized control

If the system permits transition only between a pair of "comfort" and "saver" set points, an attacker could send false "I'm coming home" messages to change set points and waste energy. If it permits arbitrarily changing set points, the attacker could subject the house to extremes of heat and cold or even turn off the system, causing pipes to freeze in the winter and pets to die of heat in the summer.

Allowing Internet control of a thermostat gives rise to several potential attacks.

Of course, a properly designed system with safety interlocks and a well-administered password policy could prevent this from happening. But the potential for it to occur makes it a threat that must be countered.

Internet thermostats also offer utility companies the possibility of suggesting or demanding changes in thermostat operation during periods of peak demand. Some US electric power companies already use radio commands to disable or reduce the duty cycle of air conditioning units during peak loads. Utility customers volunteer to do this and are compensated for the inconvenience.

Internet thermostats can make this process more sophisticated. The utility could instruct each thermostat to change its set point a few degrees to ease power requirements during peak loads. Because air conditioners form a significant part of peak electricity demand during hot summer days, this mechanism could make the difference between a blackout and continued operation of the power grid.

But such a system creates potential vulnerabilities. For example, someone might trick a number of thermostats into thinking that it is not a peak day, thereby increasing demand. If done on a broad enough scale, this could cause power-grid failure, especially if the electricity provider has factored the ability to change set points into its plan for sizing its generating capacity.

Centralized control for a power-saving scheme can create even more serious problems. Attacks that successfully break into the central control computer for set point commands, or even just spoof commands, can attempt to coordinate power consumption among many homes.

What if someone wrote a virus that took over computers for the purpose of launching attacks on all Internet-connected thermostats? The approach could be subtle, such as bumping thermostat temperature a bit hotter in the winter or cooler in the summer to increase energy consumption and inflate utility bills.

Or it could be not so subtle: If all the thermostats in a city suddenly activated their air conditioners simultaneously during a peak load period, the power surge could cause a significant problem.

Then there are pranks. What if some kid on the other side of the world decided to change your thermostat setting by 20 degrees while you were asleep every night? (Actually, using your home control system to flash your lights on and off might be more entertaining, but we're talking about thermostats.)

Battery attacks

Many thermostats, including at least one Internet brand, are battery powered. This is partly because line voltage isn't available and partly because safely converting line voltage to a thermostat's needs takes a large converter that costs money, requires extra wiring, and complicates electrical safety.

Some thermostats use wireless networking to avoid wiring costs, but too

many networking conversations can run the battery down quickly. If the thermostat is connected to the Internet, an attacker could run the battery down simply by repeatedly querying the thermostat's status.

A low-voltage detection circuit could disable the wireless connection before the battery died, but the developer needs to design this capability into the system.

Privacy

A person who can monitor your thermostat setting could also determine whether you're likely to be asleep, at home, or out of the house. Even if an attacker can't query the thermostat directly, simply monitoring traffic for inbound packets talking to the thermostat can indicate whether the house is vacant—and a potential burglary target.

An Internet-enabled thermostat can also let Big Brother monitor whether you're setting it properly to do your part for an energy crisis—and set it for you if you're not. Some gas, water, and electric meters already report to utility companies via modem, so the infrastructure for automated utility monitoring is already in place.

A thermostat controls only a limited amount of energy release, and people are often around to notice it's misbehaving. Other application areas are more challenging—for example, connecting vehicles to the Internet.

In many ways, we aren't ready to deal with the security challenges we are sure to face. Some involve simply ensuring that design teams acquire the right skills as they start making products that are exposed to security risks, but others involve significant research before we can hope to address them.

For example, how do we create impenetrable firewalls to keep attackers from manipulating safety-critical sensors and actuators? How can we ensure that real-time deadlines will be met, even in the face of DoS attacks or compromised system components? Can we create intrusion-detection systems that can respond fast enough to restore a system to correct operation before a 50-millisecond control loop loses stability?

Can we securely upgrade unattended embedded systems without being vulnerable to attacks on the upgrading mechanism? Can we detect and avoid attacks designed to drain batteries? Can we do all this on a $1 microcontroller?

We have made progress on some of these problems, of course. But some important areas aren't on enterprise-focused research agendas, and the need for embedded security is already upon us. ■

Philip Koopman *is an associate professor in the Department of Electrical and Computer Engineering at Carnegie Mellon University, where he is also a member of the Institute for Complex Engineered Systems and the Institute for Software Research International. Contact him at koopman@cmu.edu.*

In Defense of PowerPoint

Neville Holmes, University of Tasmania

Some minor computing issues become major when repeatedly made public. The denigration of PowerPoint is such an issue, one that, like influenza, seems to come in seasonal waves.

The most recent wave started in the US as "PowerPoint Is Evil" (*Wired*, Sept. 2003; www.wired.com/wired/archive/11.09/ppt2.html), spread to Australia as "Death by Slides" (*Financial Review*, 15 Nov. 2003; http://afr.com/articles/2003/11/14/1068674378566.html), to the UK as "How PowerPoint Can Fatally Weaken Your Argument" (*The Observer*, 21 Dec. 2003; http://observer.guardian.co.uk/business/story/0,6903,1110963,00.html), and back to the US as "Does PowerPoint Make Us Stupid?" (*CNN*, 30 Dec. 2003; www.cnn.com/2003/TECH/ptech/12/30/byrne.powerpoint.ap/). Further, a *Non Sequitur* cartoon this past April 29th showed a fully equipped Environmental Protection Agency squad storming triumphantly into a PowerPoint presentation.

This kind of story is not a joke. Although the authors of such post hoc arguments show occasional appreciation that the user should bear a little of the blame, they convey the overall impression that vile PowerPoint corrupts minds.

Yet it is their argument that's corrupt. PowerPoint is no more responsible for bad presentations than chainsaws are responsible for the clear-felling of old-growth forests. Technology is not itself responsible for the uses to which it is put—technology's users must shoulder the blame.

Professionals, not the software packages they use, are responsible for the presentations they make.

Computing professionals who blame their machinery for their failures set a bad example for computer users already prone to using the computer as a scapegoat. Countering silly arguments about PowerPoint requires a full appreciation of presentation technology and of the uses to which it might be put. PowerPoint is just presentation technology's latest iteration and will eventually be replaced by something else.

HISTORY LESSON

Presentation technology first took a *direct* form. In Europe more than two millennia ago, presenters developed mnemonic techniques. For centuries, early books served only as a reference for presentations, the idea of reading silently being considered strange when first introduced.

Later, chalk and blackboards served as standard equipment in classrooms, with fancy blackboards being used in lecture rooms to record up to an hour or two of lecturing. Then came felt-tip or marker pens, flip charts, and the whiteboard.

The first machine I remember being used for *indirect* presentation, apart from the movie projector, was the *epidiascope*, a rather cumbersome machine that projected by reflected light. The development of robust transparent foolscap sheets made the overhead projector popular, although it could also be used with transparent scrolls. Presenters also occasionally used 35-mm photographic slides, particularly at conferences, but the projectors needed an operator and thus were rather "accident prone."

In the early 1970s, the availability of television sets brought computers into use for presentations. The first such machine I used—the 5100, IBM's second-generation personal computer—had a video socket at the back for use in connecting it by coaxial cable to a classroom TV. Later versions of the PC required more complex connections to display the panels that replaced the traditional transparencies sitting atop the overhead projector. Nowadays, PCs usually are connected to more powerful machines that double as video projectors.

As technology has advanced and the market expanded, developers have crafted a great variety of software. Early on, I found that I could easily use a debug script for DOS to place text anywhere I wanted on the screen, and I used batch files to put together and control presentations overall. I only switched to PowerPoint, which I found very difficult to make do what I wanted, when projectors could no longer be relied on to properly project 40-column DOS screens.

Continued on page 98

Continued from page 100

USES

Although presentations take many forms, they all combine three independent motivations, much as hues can be depicted as located within a color triangle according to their three basic components. Presentations can be used to

- convey information,
- collect information, and
- persuade an audience.

Currently, presenters most neglect the persuasive aspect, yet in olden times, knowledge of rhetorical principles was considered one of a classical education's more important benefits. The persuasive aspect is also the most significant for professionals, who must use facts and reasoning to help their clients and audiences make good decisions.

Informing

Education at all levels focuses in part on getting facts and ideas across to people. The professional issue involves determining what combination of technology and technique will do this best. Some teachers see PowerPoint as a splendid tool to help them convey ideas. Others prefer to use browser-driven HTML.

With the digital technology now available, we must ask if face-to-face presentation offers the best way to inform students. In the classroom at least, students learn better by doing than by merely listening or reading. I still remember a time early in my schooling when I spent hours and hours writing answers to sums on a slate. Thanks to this instruction, I can still sometimes astonish young shop assistants by giving them the correct sum long before their cash register tells them what the total is.

Our digital technology would be better used in the classroom by administering drill and practice as a foundation for literacy and numeracy so that teachers can concentrate on the more important job of inculcating and encouraging social capability, which they must do personally. Using computer-based academic gaming to create social situations could greatly help them in accomplishing this task.

But none of this involves PowerPoint. PowerPoint does, however, reign supreme at conferences—although conference presenters who merely recapitulate the contents of their paper place themselves in danger of putting their audience to sleep, particularly straight after lunch. A more effective approach uses the presentation to persuade the majority of the audience to actually read the paper, then devotes the question time to those few who have *already* read it. All of which puts conference presentations much more in the third class: persuasion.

Some teachers see PowerPoint as a splendid tool to help them convey ideas.

Gathering data

Presentations are given in person—which means a successful presentation must be interactive. A good presenter will maintain eye contact and will, even in the absence of questions from his listeners, pick up their reactions and modify the presentation accordingly. A canned presentation, however, is hard to modify, whatever the software used. The more intricate the data, the harder the modification.

Many computing professionals focus on data gathering, which can often be done most effectively by taking a group of key informants away from their day-to-day activities to quiz them. These sessions need skilled management. The lead computing professional will typically start off with a presentation that gradually merges into a controlled group discussion. PowerPoint would be inappropriate here.

Although many professionals now prefer using electronic whiteboards in brainstorming and similar activities, I find it difficult to believe that the traditional marker pen and flip chart approach wouldn't be better. The atmosphere in a room with scribbled flip charts hung up around the walls, with new charts being added, old ones being revised and re-sorted, and people moving around and discussing them, can be exhilarating and highly productive.

Persuading

The most important events in a computing professional's career involve the formal presentations given to persuade clients to accept a proposal or the results of work done. Here PowerPoint comes into its own.

Although some data must be shown, the focus stays on the presenter, not the PowerPoint slides. The slides should be as simple and undistracting as possible. Presenters need two kinds of slides for persuasive presentations: bullet-point slides and data slides.

Long ago, I used my DOS batch files with one overhead projector for the bullet points and used transparencies on another overhead projector for the data slides, with a screen for each. This proved more effective than would using PowerPoint on a single projector today.

Data slides must be simplified so that their meaning becomes apparent only as the presenter explains them. Too much detail usually distracts audience members from the presenter and annoys them with difficult-to-read fine print. If presenters plan to show a lot of fancy graphics, they would probably be safer and better off simply showing them all from a videotape or DVD.

Bullet points—PowerPoint's most vilified aspect—are the most misunderstood of presentation techniques. As far as the audience is concerned, bullet points only serve to remind them of the presentation's general context. As far as the presenter is concerned, bullet points replace the mnemonic techniques handed down from the Greeks and the more recent prompt cards hidden in the hand of a formal debater.

Bullet-point slides should be as simple as possible, especially in content.

This simplicity provides two important benefits: It lessens the distraction to the audience and supports spontaneity in the presenter—which is even more important than maintaining eye contact. Simple text lets presenters more easily lengthen or shorten the presentation to fit the time allowed, and it also lets them use larger letters.

Spontaneity can be achieved by rehearsing what might be said in respect to each bullet point so that when the point turns up in the presentation, the audience will see that the presenter is actively choosing what to say. The only thing that makes a worse impression on an audience than reciting from memory is reading from the screen—especially because the audience can read the text faster silently than the presenter can read it aloud. If a presenter must put up a long quotation, it should be a data slide that's easy for the audience to read.

A sans serif font is more legible onscreen than a serif one. I also suggest white letters on a black background.

For a computing professional, being able to give a good presentation is essential. Professional training should thus include instruction and practice in making presentations of all kinds, with the objective not so much to constrain the budding professional to any particular set of rules as to emphasize the importance of skill in presentations and purposeful thought in preparing for them.

Students should be trained in determining whether presentations are appropriate in different professional situations and in designing different kinds of presentations for different circumstances. In many situations, simple face-to-face discussions offer the best approach.

Presentations raise a much broader issue for all computing professionals, however. All too often, commentators and authors outside the profession—and sadly some within—take an irrational stand on digital technology, blaming it for all kinds of social and economic ills. The condemnation of PowerPoint is only an obvious example. We must be sensitive to errors of this kind in ourselves and loud in counteracting public errors of this kind in others. ■

Neville Holmes *is an honorary research associate at the University of Tasmania's School of Computing. Contact him at neville.holmes@utas.edu.au. Details of citations in this essay, and links to further material, are at www.comp.utas.edu.au/users/nholmes/prfsn.*