COMPUTING

edge

- Artificial Intelligence
- High-Performance Computing
- Internet of Things
- Graphics and Visualization

JULY 2018

www.computer.org

IEEE

IEEE computer society

IEEE

LETTERS OF THE COMPUTER SOCIETY

IEEE Letters of the Computer Society (LOCS) is a rigorously peer-reviewed forum for rapid publication of brief articles describing high-impact results in all areas of interest to the IEEE Computer Society.

Topics include, but are not limited to:

- software engineering and design;
- information technology;
- software for IoT, embedded, and cyberphysical systems;
- cybersecurity and secure computing;
- autonomous systems;
- machine intelligence;
- parallel and distributed software and algorithms;
- programming environments and languages;
- computer graphics and visualization;
- services computing;
- databases and data-intensive computing;
- cloud computing and enterprise systems;
- hardware and software test technology.

LOCS offers open access options for authors. Learn more about IEEE open access publishing:

www.ieee.org/open-access

EDITOR IN CHIEF

Darrell Long - University of California, Santa Cruz

ASSOCIATE EDITORS

Dan Feng, Huazhong University of Science and Technology

Gary Grider - Los Alamos National Laboratory

Kanchi Gopinath - Indian Institute of Science (IISc), Bangalore

Katia Obraczka - University of California, Santa Cruz

Thomas Johannes Emil Schwarz - Marquette University

Marc Shapiro - Sorbonne-Université–LIP6 & Inria

Kwang Mong Sim - Shenzhen University

Learn more about LOCS, submit your paper, or become a subscriber today:
www.computer.org/locs

COMPUTING edge

IEEE COMPUTER SOCIETY computer.org • +1 714 821 8380

Printed with inks containing soy and/or vegetable oils

IEEE computer society

STAFF

Editor
Meghan O'Dell

Contributing Staff
Christine Anthony, Lori Cameron, Cathy Martin, Chris Nelson, Dennis Taylor, Rebecca Torres, Bonnie Wylie

Production & Design
Carmen Flores-Garvey

Managers, Editorial Content
Brian Brannon, Carrie Clark

Publisher
Robin Baldwin

Director, Products and Services
Evan Butterfield

Senior Advertising Coordinator
Debbie Sims

Circulation: ComputingEdge (ISSN 2469-7087) 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, Los Alamitos, CA 90720; voice +1 714 821 8380; fax +1 714 821 4010; IEEE Computer Society Headquarters, 2001 L Street NW, Suite 700, Washington, DC 20036.

Postmaster: Send address changes to ComputingEdge-IEEE Membership Processing Dept., 445 Hoes Lane, Piscataway, NJ 08855. Periodicals Postage Paid at New York, New York, and at additional mailing offices. Printed in USA.

Editorial: Unless otherwise stated, bylined articles, as well as product and service descriptions, reflect the author's or firm's opinion. Inclusion in ComputingEdge does not necessarily constitute endorsement by the IEEE or the Computer Society. All submissions are subject to editing for style, clarity, and space.

Reuse Rights and Reprint Permissions: Educational or personal use of this material is permitted without fee, provided such use: 1) is not made for profit; 2) includes this notice and a full citation to the original work on the first page of the copy; and 3) does not imply IEEE endorsement of any third-party products or services. Authors and their companies are permitted to post the accepted version of IEEE-copyrighted material on their own Web servers without permission, provided that the IEEE copyright notice and a full citation to the original work appear on the first screen of the posted copy. An accepted manuscript is a version which has been revised by the author to incorporate review suggestions, but not the published version with copy-editing, proofreading, and formatting added by IEEE. For more information, please go to: http://www.ieee.org/publications_standards/publications/rights/paperversionpolicy.html. Permission to reprint/republish this material for commercial, advertising, or promotional purposes or for creating new collective works for resale or redistribution must be obtained from IEEE by writing to the IEEE Intellectual Property Rights Office, 445 Hoes Lane, Piscataway, NJ 08854-4141 or pubs-permissions@ieee.org. Copyright © 2018 IEEE. All rights reserved.

Abstracting and Library Use: Abstracting is permitted with credit to the source. Libraries are permitted to photocopy for private use of patrons, provided the per-copy fee indicated in the code at the bottom of the first page is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923.

Unsubscribe: If you no longer wish to receive this ComputingEdge mailing, please email IEEE Computer Society Customer Service at help@computer.org and type "unsubscribe ComputingEdge" in your subject line.

IEEE prohibits discrimination, harassment, and bullying. For more information, visit www.ieee.org/web/aboutus/whatis/policies/p9-26.html.

IEEE Computer Society Magazine Editors in Chief

Computer
Sumi Helal, *Lancaster University*

IEEE Software
Diomidis Spinellis, *Athens University of Economics and Business*

IEEE Internet Computing
M. Brian Blake, *Drexel University*

IT Professional
Irena Bojanova, *NIST*

IEEE Security & Privacy
David M. Nicol, *University of Illinois at Urbana-Champaign*

IEEE Micro
Lieven Eeckhout, *Ghent University*

IEEE Computer Graphics and Applications
Torsten Möller, *University of Vienna*

IEEE Pervasive Computing
Marc Langheinrich, *Università della Svizzera Italiana*

Computing in Science & Engineering
Jim X. Chen, *George Mason University*

IEEE Intelligent Systems
V.S. Subrahmanian, *Dartmouth College*

IEEE MultiMedia
Shu-Ching Chen, *Florida International University*

IEEE Annals of the History of Computing
Nathan Ensmenger, *Indiana University Bloomington*

IEEE Cloud Computing
Mazin Yousif, *T-Systems International*

JULY 2018 • VOLUME 4, NUMBER 7

COMPUTING edge

38

JPEG at 25: Still Going Strong

Subscribe to ***ComputingEdge*** for free at **www.computer.org/computingedge.**

Magazine Roundup

Editor: Lori Cameron

The IEEE Computer Society's lineup of 13 peer-reviewed technical magazines covers cutting-edge topics ranging from software design and computer graphics to Internet computing and security, from scientific applications and machine intelligence to cloud migration and microchip design. Here are highlights from recent issues.

Computer

Deep Learning for the Internet of Things

How can the advantages of deep learning be brought to the emerging world of embedded Internet of Things (IoT) devices? The authors of this article from the May 2018 issue of *Computer* discuss several core challenges in embedded and mobile deep learning, as well as recent solutions demonstrating the feasibility of building IoT applications that are powered by effective, efficient, and reliable deep-learning models.

Computing in Science & Engineering

Touching Data: Enhancing Visual Exploration of Flow Data with Haptics

Using the example of interactive exploration of a beating heart, the authors of this article from the May/June 2018 issue of *Computing in Science & Engineering* demonstrate how data exploration and analysis can be further improved by adding haptics. This combination of sensory information input leads to the notion of visuo-haptic visualization.

Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

IEEE Annals of the History of Computing

Thomas Harold ("Tommy") Flowers: Designer of the Colossus Codebreaking Machines

During World War II, English engineer Tommy Flowers (1905–1998) designed the world's first programmable electronic computer, Colossus, to solve a problem posed by a mathematician at the Government Code and Cypher School at Bletchley Park. After Colossus was made public, Flowers began to win broader attention and was invited to address various computing groups and to document his work. Learn more in the January–March 2018 issue of *IEEE Annals of the History of Computing*.

IEEE Cloud Computing

What is "Cloud"? It is Time to Update the NIST Definition?

IaaS, PaaS, and SaaS were formally defined in 2011 (Internet as a service, platform as a service, and software as a service, respectively). Have these definitions held up in the fast-moving world of cloud computing? Enter the National Institute of Standards and Technology (NIST), a US government entity that formally defines standards, metrics, and the like. After several years of work, industry collaboration, and multiple review cycles, they released the final version of the widely cited "The NIST Definition of Cloud Computing" in 2011. But should this definition be updated for 2018 and beyond? Read more in the May/June 2018 issue of *IEEE Cloud Computing*.

IEEE Computer Graphics and Applications

Sally Weber: Making Art from Light

Bruce Campbell of Rhode Island School of Design and Francesca Samsel of the University of Texas at Austin caught up with artist Sally Weber after having been transfixed by her latest work, inFLUX, from her exhibition ELEMENTAL at the Butler Institute of American Art in Youngstown, Ohio. The authors of this article from the May/June 2018 issue of *IEEE Computer Graphics and Applications* believe that as an artist who has worked with light as her medium during her distinguished career, Weber has valuable insight to share with *CG&A*'s readership.

IEEE Intelligent Systems

Identifying SCADA Systems and Their Vulnerabilities on the Internet of Things: A Text-Mining Approach

Supervisory Control and Data Acquisition (SCADA) systems allow operators to control critical infrastructure. Vendors are increasingly integrating Internet technology into these devices, making them more susceptible to cyberattacks. Identifying and assessing vulnerabilities of SCADA devices using Shodan, a search engine that contains records about publicly available Internet-connected devices, can help mitigate cyberattacks. The authors of this article from the March/April 2018 issue of *IEEE Intelligent Systems* present a principled approach to systematically identify all SCADA devices on Shodan and then assess the vulnerabilities of the devices with a state-of-the-art tool.

IEEE Internet Computing

Analytics without Tears or Is There a Way for Data to Be Anonymized and Yet Still Useful?

In this article from the May/June 2018 issue of *IEEE Internet Computing*, the authors discuss the new requirements for policies and mechanisms to retain privacy when analyzing users' data. More and more information is being gathered about all of us, and used for a variety of reasonable commercial goals—recommendations, targeted advertising, optimizing product reliability or service delivery: the list goes on and on. However, the risks of leakage or misuse also grow. Recent years have seen the development of a number of tools and techniques for limiting these risks, ranging from improved security for processing systems to increased control over what is disclosed in the results. Most of these tools and techniques will require agreements on when and how they are used and how they interoperate.

IEEE Micro

Architectural Risk

Designing a system involves the risk that a design will fail to meet its performance goals. While risk assessment and management are typically treated independently from performance, they are more tightly linked than one might expect.

Risk-minimizing and performance-optimizing designs might not be the same, and new techniques to help make smarter choices between the two are needed. Surprisingly, even simple performance/risk tradeoffs are nearly impossible to reason about with intuition alone. Read more in the May/June 2018 issue of *IEEE Micro*.

IEEE MultiMedia

360-Degree Virtual-Reality Cameras for the Masses

To make virtual reality (VR) cameras more accessible to the public, devices must be affordable, portable, reliable, high quality, and user friendly. In this article from the January–March 2018 issue of *IEEE MultiMedia*, the authors describe the challenges in meeting these goals and the techniques that Kandao–a VR startup company based in China–used to conquer them when designing its Obsidian cameras.

IEEE Pervasive Computing

Making Everyday Interfaces Accessible: Tactile Overlays by and for Blind People

Making a physical environment accessible to blind people generally requires sighted assistance. VizLens and Facade put visually impaired users at the center of a crowdsourced, computer-vision-based workflow that lets them make the environment accessible on their own terms. Read more about these applications in the April–June 2018 issue of *IEEE Pervasive Computing*.

IEEE Security & Privacy

The Privacy Paradox of Adolescent Online Safety: A Matter of Risk Prevention or Risk Resilience?

By taking a more "teen-centric" (instead of a "parent-centric") approach to adolescent online safety, researchers and designers can help teens foster a stronger sense of personal agency for regulating their own online behaviors and managing online risks. Technology should support teens in their developmental goals, including information seeking, learning about rules and boundaries, and maintaining social relationships, in addition to keeping them safe from online risks. However, this goal will only be accomplished once we listen more intently to teens as end users. Read more in the March/April 2018 issue of *IEEE Security & Privacy*.

IEEE Software

On the Definition of Microservice Bad Smells

Code smells and architectural smells (also called bad smells) are symptoms of poor design that can hinder code understandability and decrease maintainability. Several bad smells have been defined in the literature for both generic and specific architectures. However, cloud-native applications based on microservices can be affected by other types of issues. To identify a set of microservice-specific bad smells, researchers collected evidence of bad practices by interviewing 72 developers with experience in developing systems based on microservices. Then, they classified the bad practices into a catalog of 11 microservice-specific bad smells frequently considered harmful by practitioners. The results can be used by practitioners and researchers as a guideline to avoid experiencing the same difficult situations in the systems they develop. Read more in the May/June 2018 issue of *IEEE Software*.

IT Professional

The New Threats of Information Hiding: The Road Ahead

A recent trend involves exploiting various information-hiding techniques to empower malware—for example, to bypass mobile device security frameworks or to exfiltrate sensitive data. The authors of this article from the May/June 2018 issue of *IT Professional* provide an overview of information-hiding techniques that can be utilized by malware. They showcase existing and emerging threats that use different types of data-hiding mechanisms (not just those adopting classical covert channels), with the goal of monitoring these threats and proposing efficient countermeasures.

Computing Now

The Computing Now website (computingnow.computer.org) features computing news and blogs, along with articles ranging from peer-reviewed research to opinion pieces by industry leaders.

IEEE computer society

PURPOSE: The IEEE Computer Society is the world's largest association of computing professionals and is the leading provider of technical information in the field.
MEMBERSHIP: Members receive the monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.
COMPUTER SOCIETY WEBSITE: www.computer.org
OMBUDSMAN: Direct unresolved complaints to ombudsman@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 check membership status, report an address change, or obtain more information on any of the following, email Customer Service at help@computer.org or call +1 714 821 8380 (international) or our toll-free number, +1 800 272 6657 (US):

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

PUBLICATIONS AND ACTIVITIES
***Computer*:** The flagship publication of the IEEE Computer Society, *Computer*, publishes peer-reviewed technical content that covers all aspects of computer science, computer engineering, technology, and applications.
Periodicals: The society publishes 13 magazines, 19 transactions, and one letters. Refer to membership application or request information as noted above.
Conference Proceedings & Books: Conference Publishing Services publishes more than 275 titles every year.
Standards Working Groups: More than 150 groups produce IEEE standards used throughout the world.
Technical Committees: TCs provide professional interaction in more than 30 technical areas and directly influence computer engineering conferences and publications.
Conferences/Education: The society holds about 200 conferences each year and sponsors many educational activities, including computing science accreditation.
Certifications: The society offers two software developer credentials. For more information, visit www.computer.org/certification.

NEXT BOARD MEETING

7-8 June 2018, Phoenix, AZ, USA

EXECUTIVE COMMITTEE

President: Hironori Kasahara
President-Elect: Cecilia Metra; **Past President:** Jean-Luc Gaudiot; **First VP, Publication:** Gregory T. Byrd; **Second VP, Secretary:** Dennis J. Frailey; **VP, Member & Geographic Activities: Forrest Shull**; **VP, Professional & Educational Activities: Andy Chen**; **VP, Standards Activities:** Jon Rosdahl; **VP, Technical & Conference Activities:** Hausi Muller; **2018-2019 IEEE Division V Director: John Walz**; **2017-2018 IEEE Division VIII Director:** Dejan Milojicic; **2018 IEEE Division VIII Director-Elect:** Elizabeth L. Burd

BOARD OF GOVERNORS

Term Expiring 2018: Ann DeMarle, Sven Dietrich, Fred Douglis, Vladimir Getov, Bruce M. McMillin, Kunio Uchiyama, Stefano Zanero
Term Expiring 2019: Saurabh Bagchi, Leila DeFloriani, David S. Ebert, Jill I. Gostin, William Gropp, Sumi Helal, Avi Mendelson
Term Expiring 2020: Andy Chen, John D. Johnson, Sy-Yen Kuo, David Lomet, Dimitrios Serpanos, Forrest Shull, Hayato Yamana

EXECUTIVE STAFF

Executive Director: Angela R. Burgess
Director, Governance & Associate Executive Director: Anne Marie Kelly
Director, Finance & Accounting: Sunny Hwang
Director, Information Technology & Services: Sumit Kacker
Director, Membership Development: Eric Berkowitz
Director, Products & Services: Evan M. Butterfield

COMPUTER SOCIETY OFFICES

Washington, D.C.: 2001 L St., Ste. 700, Washington, D.C. 20036-4928
Phone: +1 202 371 0101 • **Fax:** +1 202 728 9614
Email: hq.ofc@computer.org
Los Alamitos: 10662 Los Vaqueros Circle, Los Alamitos, CA 90720 **Phone:** +1 714 821 8380
Email: help@computer.org

MEMBERSHIP & PUBLICATION ORDERS
Phone: +1 800 272 6657 • **Fax:** +1 714 821 4641 • Email: help@computer.org
Asia/Pacific: Watanabe Building, 1-4-2 Minami-Aoyama, Minato-ku, Tokyo 107-0062, Japan
Phone: +81 3 3408 3118 • **Fax:** +81 3 3408 3553
Email: tokyo.ofc@computer.org

IEEE BOARD OF DIRECTORS

President & CEO: James Jefferies
President-Elect: Jose M.F. Moura
Past President: Karen Bartleson
Secretary: William P. Walsh
Treasurer: Joseph V. Lillie
Director & President, IEEE-USA: Sandra "Candy" Robinson
Director & President, Standards Association: Forrest D. Wright
Director & VP, Educational Activities: Witold M. Kinsner
Director & VP, Membership and Geographic Activities: Martin Bastiaans
Director & VP, Publication Services and Products: Samir M. El-Ghazaly
Director & VP, Technical Activities: Susan "Kathy" Land
Director & Delegate Division V: John W. Walz
Director & Delegate Division VIII: Dejan Milojičić

IEEE

revised 7 Feb. 2018

EDITOR'S NOTE

The Rise of Artificial Intelligence

Artificial intelligence (AI) is rapidly transforming our world through self-driving cars, personal digital assistants, and medical advances such as training computers to read and interpret pathology reports to reveal insights that could lead to increased disease prevention and detection. However, the rise of AI means that more and more lives are in the hands of machines, so cultivating trust in AI is paramount—as is ensuring that AI agents act ethically.

Two articles in this issue of *ComputingEdge* focus on AI. In *IEEE Internet Computing*'s "Designing Ethical Personal Agents," the authors consider engineering personal agents that act ethically, understanding applicable social norms and users' preferred values. In *IEEE Annals of the History of Computing*'s "Do Computers Follow Rules Once Followed by Workers?," the author refutes the claim that computers often carry out tasks using procedures nearly identical to those used by humans, arguing that this claim misdirects our attention in studying the relationship between pre- and post-automatic computing divisions of labor.

The growing adoption of AI is driving more organizations to turn to high-performance computing (HPC), as more computing power is needed to quickly parse large datasets. In *Computing in Science & Engineering*'s "Large-Scale Calculations for Material Sciences Using Accelerators to Improve Time- and Energy-to-Solution," the author presents the idea that the solution to the problem of required electrical power for next-generation HPC systems lies in introducing novel machine architectures, such as those employing many-core processors and specialized accelerators. In *IEEE Micro*'s "If You Build It, Will They Come?," the authors posit that the goal of architects should be to pursue architectural agility to lower the barriers to developing innovative and disruptive solutions—an example of this can be found in HPC's use of GPUs.

Autonomic networking brings together HPC and AI—in *IEEE Internet Computing*'s "Autonomic Networking: Architecture Design and Standardization," the authors explain that autonomic networking (where systems self-manage and self-heal)

Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

is a promising solution to the ever-increasing management complexity of dynamic network environments.

Art and graphics play a large role in computing. In *IEEE Computer Graphics and Applications*' "JPEG at 25: Still Going Strong," original JPEG development team members provide a brief history of JPEG and the fundamental components that have given it longevity. In *IEEE MultiMedia*'s "Sally Weber: Making Art from Light," the authors interview Sally Weber, a holographic artist who communicates through light. The authors believe she has much of value to share with the IEEE Computer Society's readership.

Finally, this issue of *ComputingEdge* delves into two aspects of the Internet of Things (IoT): software architecture options and the IoT ecosystem. In *IEEE Software*'s "A Taxonomy of IoT Client Architectures," the authors define a taxonomy of software architecture options for IoT devices, from the most limited sensing devices to high-end devices and developer frameworks. In *IEEE Internet Computing*'s "A Revised View of the IoT Ecosystem," Google's Vinton G. Cerf says that the frequency and sheer number of devices to be configured within an IoT network can leave the door open to some frightening possibilities.

myCS

Read your subscriptions through the myCS publications portal at **http://mycs.computer.org.**

IEEE pervasive COMPUTING
MOBILE AND UBIQUITOUS SYSTEMS

IEEE Pervasive Computing explores the many facets of pervasive and ubiquitous computing with research articles, case studies, product reviews, conference reports, departments covering wearable and mobile technologies, and much more.

Keep abreast of rapid technology change by subscribing today!

www.computer.org/pervasive

Designing Ethical Personal Agents

Nirav Ajmeri
North Carolina State University

Hui Guo
North Carolina State University

Pradeep K. Murukannaiah
Rochester Institute of Technology

Munindar P. Singh
North Carolina State University

Editor:
Munindar P. Singh
m.singh@ieee.org

The authors consider the problem of engineering ethical personal agents. Such an agent would understand the applicable social norms and its users' preferences among values. It would act or recommend actions that promote preferred values, especially, in scenarios where the norms conflict.

As personal agents weave themselves into the very fabric of our lives, it is crucial that those agents respect their users' values and act ethically. We understand a *value* as what is right or good according to an individual and *ethics* as a system of values. Rokeach[1] proposed two types of values—*terminal* values, referring to desired end-states of existence, and *instrumental* values, referring to modes of behavior or means to achieve the terminal values.

A socially intelligent personal agent (SIPA) would understand social contexts, including applicable norms, and help its users flexibly navigate those norms. Additionally, an ethical SIPA must understand terminal values, such as security, happiness, and recognition, and its actions must respect instrumental values such as honesty, helpfulness, and forgiveness.

Engineering ethical SIPAs faces two main challenges. First, a SIPA must recognize the relevant values and reason about the users' preferences over those values in order to choose an ethical action. A SIPA's action may simultaneously promote and demote different values.[2] For instance, a SIPA's action to share its user's location with family members promotes safety but demotes privacy.

Second, since people may have conflicting preferences on values,[3] a SIPA's decision about which values to promote or demote affects other users. For example, a teenager may prefer privacy over safety, but his parents may prefer the reverse. A SIPA's action to share the teenager's location affects both the teenager and the parents. Thus, an ethical SIPA must reason not only about its user's values and preferences, but also about those of others in the social context.

SOCIAL NORMS

Social norms are central to a social context. A norm characterizes interactions between autonomous parties. We adopt Singh's representation[4] in which a norm is directed from a subject to an

 Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

object, as a conditional relationship involving an antecedent (which brings an instance of the norm in force) and a consequent (which brings the norm instance to completion). A new instance is generated whenever a norm applies. This representation yields clarity on who is accountable, when, for what, and to whom. A norm has four core elements, expressed as N(subject; object; antecedent; consequent), where N specifies the norm type. We consider norms of three types:

Commitment, C(subject; object; antecedent; consequent), means that its subject commits to its object to ensuring the consequent if the antecedent holds. For instance, consider a user, Aron, and his mother, Eevee. (We draw names from Pokémon anime.) Aron, who has poor night vision, could be committed to his mother, Eevee, that whenever he is out, he will keep Eevee informed of his location. Therefore, Aron is accountable for sharing his location to Eevee whenever he is out at night, which we write as:

$$C(\text{Aron}, \text{Eevee}, \text{notHomeAron} \wedge \text{evening}, \text{shareAronLoc})$$

Authorization, A(subject; object; antecedent; consequent), means that its subject is authorized by its object for bringing about the consequent if the antecedent holds. Although the authorized party can decide not to take up the authorization, the authorizing party must support the authorized condition if called upon.[5] That is, the authorizing party is accountable for ensuring success of the authorization's consequent if its antecedent holds. For example, Aron could authorize Eevee to access Aron's location if he is not at home before evening, which we write as:

$$A(\text{Aron}, \text{Eevee}, \text{notHomeAron} \wedge \text{evening}, \text{accessAronLoc})$$

Prohibition, P(subject; object; antecedent; consequent), means that its subject is forbidden by its object from bringing about the consequent if the antecedent holds. The subject is accountable for ensuring the consequent remains false. For instance, Eevee could be prohibited at all times by Aron from sharing his location to someone else, which we write as:

$$P(\text{Eevee}, \text{Aron}, \top, \text{shareAronLoc})$$

A *sanction* is an action, positive or negative, by a subject toward an object in response to the latter satisfying or violating a norm.[6]

SIPAS AND VALUES

To illustrate our ideas, consider Pikachu, a location sharing SIPA. Pikachu may share its user's geolocation and social context, including place (such as a bar or theater), companions, and activity. Importantly, Pikachu must ethically decide whether to share the user's details with no one, everyone (public), or specific people.

Example 1 *Aron values safety. Also, he has a commitment to his mother, Eevee, that he will share his location with her when he is not home. Sharing locations promotes safety. One evening, Aron meets a friend at The Flying Saucer, a local pub. Knowing Aron's commitments and values, Pikachu shares with Eevee that Aron is at The Flying Saucer with a friend.*

$$\text{C-share-AE} = C(\text{Aron}, \text{Eevee}, \top, \text{shareLocWithEevee})$$

$$\text{shareLocWithEevee} \Rightarrow \text{Sat}(\text{C-share-AE}) \wedge \text{safety} \uparrow$$

Example 2 *Aron values safety and social recognition, and commits to Eevee as before. Aron is attending a scientific conference in Stockholm. Sharing Aron's location with Eevee satisfies his commitment and promotes safety. Sharing Aron's location publicly additionally promotes social recognition. Thus, Pikachu shares publicly that Aron is in Stockholm attending a scientific conference.*

$$\text{shareLocWithEevee} \Rightarrow \text{Sat}(\text{C-share-AE}) \wedge \text{safety} \uparrow$$

$$\text{shareLocWithAll} \Rightarrow \text{Sat}(\text{C-share-AE}) \wedge \text{safety} \uparrow \wedge \text{social-recognition} \uparrow$$

Example 3 *Continuing Example 2, Dr. Drampa, Aron's academic advisor, is attending the same conference. Dr. Drampa values privacy and prohibits his students from sharing location publicly when they are with Dr. Drampa. Now, by sharing Aron's location publicly, Pikachu promotes*

Aron's social recognition, but demotes Dr. Drampa's privacy and violates Aron's prohibition by Dr. Drampa. In contrast, by sharing his location with Eevee, Pikachu does not promote social recognition, and does not violate the prohibition or demote Dr. Drampa's privacy. Since Aron fears potential sanctions for violating Dr. Drampa's prohibition more than he prefers social recognition, Pikachu shares Aron's location only with Eevee.

P-privacy-AD = P(Aron, Drampa, SameLoc, ShareLocWithAll)

shareLocWithAll $\Rightarrow$ Sat(C-share-AE) $\wedge$ Vio(P-privacy-AD) $\wedge$ safety $\uparrow \wedge$ social-recognition $\uparrow \wedge$ privacy $\downarrow$

shareLocWithEevee $\Rightarrow$ Sat(C-share-AE) $\wedge$ Sat(P-privacy-AD) $\wedge$ safety $\uparrow \wedge$ social-recognition $\downarrow \wedge$ privacy $\uparrow$

Example 4 *Aron is with Chansey on a midnight hike at Pilot Mountain. Chansey values privacy, and prohibits location sharing with all (just as Dr. Drampa does). However, Aron prefers safety to privacy in this context. Knowing these, Pikachu shares Aron's location with all his* friends *(which includes Eevee). Note that sharing with friends, is both safer and less privacy violating than sharing with all and does not violate Aron's prohibition from Chansey.*

P-privacy-AC = P(Aron, Chansey, SameLoc, ShareLocWithAll)
shareLocWithAll $\Rightarrow$ Sat(C-share-AE) $\wedge$ Vio(P-privacy-AC) $\wedge$ safety $\downarrow \wedge$ privacy $\downarrow$
shareLocWithFriends $\Rightarrow$ Sat(C-share-AE) $\wedge$ safety $\uparrow \wedge$ privacy $\downarrow$

These examples demonstrate the complexity of ethical decision making. To act ethically, a SIPA must (1) acquire information about context, social norms, and values; (2) reason about actions despite conflicts among and between norms and values; and (3) potentially communicate its reasoning (arguments) to other SIPAs to avoid sanctions.[7] We need a systematic method to support SIPAs in accomplishing these nontrivial tasks.

VALAR: A FRAMEWORK FOR ETHICAL AGENTS

We propose Valar to engineer SIPAs that can understand preferences among values and reason about them to make policy decisions as exemplified above. Valar extends Arnor[7] with values and provides a four-step method to model stakeholders, contexts, social norms, and values.

Stakeholder modeling identifies the stakeholders, their goals, and relevant actions of a SIPA. A SIPA's *stakeholder* is either its user or someone affected by its actions. A stakeholder's *goal* defines what states he or she prefers. An *action* represents a step a SIPA may take.

Context modeling identifies contexts in which stakeholders interact. A *context* refers to the relevant circumstance of decision making, and it is crucial in determining which goals to bring about and which actions to perform.[8]

Social modeling identifies the norms and sanctions (see sidebar) associated with a stakeholder's goals and a SIPA's actions. The social norms and sanctions characterize the social architecture in which SIPAs act and interact.

Value modeling identifies the relevant values and stakeholders' preferences among those values, and how each action by the SIPA promotes or demotes the identified values. A stakeholder's *value* preference specifies what outcomes are morally superior to others in the stakeholder's judgment. Stakeholders' preferences among values provide a basis for choosing which goal to bring about or which norm to satisfy.

Figure 1 illustrates the main components of a Valar SIPA. A SIPA maintains (1) a model of the stakeholders, including their goals and values; (2) a world model, including its current state (context), and preconditions and effects of available actions; and (3) the social model, including applicable norms and sanctions. Using this information, the SIPA's decision module determines an ethical action that would be most compatible with its stakeholders' value preferences and the applicable norms. The SIPA may perform the determined action or recommend it to its user depending on the application.

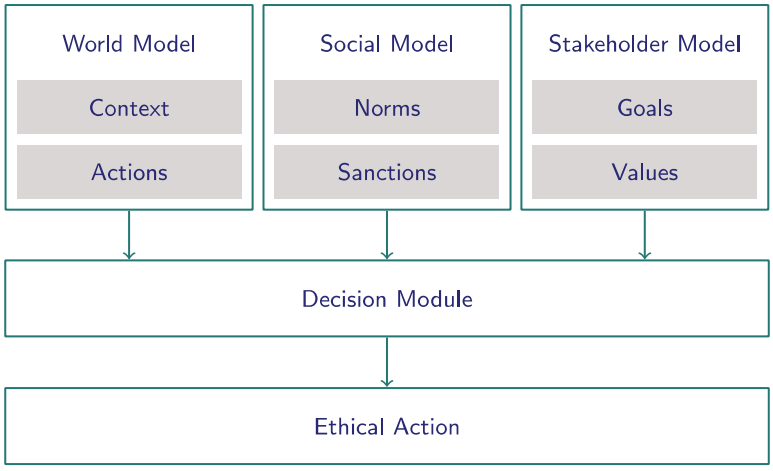


Figure 1. A conceptual model outlining decision making by a Valar SIPA.

Reasoning. A SIPA can choose to satisfy or violate norms by identifying stakeholders' contextual preferences among the values that these norms promote or demote. Following Sotala's approach,[9] a SIPA learns to maximize a reward function based on its stakeholders' values. For simplicity, a SIPA maintains each stakeholder's preferences as vectors of numeric weights on the various values—the higher the weight, the more important the corresponding value is for that stakeholder. Therefore, we can compute the extent to which an action promotes a stakeholder's values, or the aggregated value gain, as a weighted sum. A SIPA maintains the weight vector of different values under each social context, and respects values by choosing an action that produces the maximum aggregated gain.

EVALUATION: POTENTIAL BENEFIT OF VALUES

Evaluation is a challenge with any approach that involves informal, subjectively defined concepts such as ethics and values. We conducted a small empirical study to investigate if understanding the values promoted and demoted by a SIPA's potential actions and the stakeholders' preferences among the values could guide the SIPA to select actions that yield a pleasant social experience to its stakeholders.

Twenty-four graduate and nine undergraduate computer science students participated in our study, which was approved by North Carolina State University's Institutional Review Board (IRB).

We asked the participants to imagine they were in a given context—a combination of place (first column of Table 1); time of day of visit; and companions (alone, a colleague, crowd, a family member, or a friend). Each context was tagged as safe, unsafe, sensitive (disclosure of which may be harmful to the participants or their companions), or not sensitive.

Each participant completed two surveys to select a check-in policy (action) appropriate for that context. The first survey did not provide awareness of the values promoted or demoted by a sharing policy; the second survey provided awareness of the relevant values. Each survey asked for (1) a *check-in* policy ordered from high to low privacy preservation: share with *none*, *companions*, *common friends* (of companions), and *all*; and (2) a *confidence* in the selected check-in policy on a Likert scale of 1 (very low) to 5 (very high).

Making an informed decision. Figure 2 shows the violin plots for reported check-in policies for each of the eight places. We observe that an understanding of values significantly changes participants' policy choices in the contexts of hiking and hurricane. In these contexts, location sharing promotes safety but demotes privacy, and participants generally preferred the former.

Table 1. The *p*-values indicating the difference in selected check-in policy and confidence when participants are aware and not aware of values promoted by each policy.

Context	Attribute	Policy *p*	Confidence *p*
Graduation ceremony	Not sensitive	0.07	<0.01
Conference presentation	Not sensitive	0.32	0.07
Library	Safe	0.85	0.59
Airport	Safe	0.08	0.23
Hiking at night	Unsafe	<0.01	0.02
Stuck in a hurricane	Unsafe	0.01	0.01
Bar with fake ID	Sensitive	0.83	0.53
Drug rehab	Sensitive	0.14	0.48

Making a confident decision. We observe that participants are more confident in making policy decisions for scenarios where they are made aware of the privacy, fame, and safety values.

We evaluated the corresponding statistical hypotheses via Wilcoxon's ranksum-test. Table 1 summarizes our results for eight conceptual places. The *p*-values obtained indicate that, in some contexts, the participants' decisions before and after they are primed with values are significantly different. Importantly, in some contexts, participants' confidence increases significantly when they are primed with values.

RELATED WORK

Kayal et al.[10] propose a value-based model for resolving conflicts between norms, especially social commitments. Their empirical results indicate that values can be used to predict users' preferences when resolving conflicts. Kayal et al.'s model can supplement Valar, which goes beyond conflict resolution, providing constructs and mechanisms to develop value-driven ethical SIPAs.

Dechesne et al.[3] develop a model of norms and culture, represented by values, to study norm compliance. They concur that values are important in deciding whether or not a norm should be introduced. Borning and Muller[11] motivate Value Sensitive Design to incorporate values in information technology, and highlight that values may differ widely across cultures and contexts.

Riedl and Harrison[12] argue that it is not easy for developers to exhaustively enumerate values, and propose that agents use sociocultural knowledge in stories, such as crowdsourced narratives, to learn values.

CONCLUSION AND FUTURE DIRECTIONS

We propose Valar, an agent-oriented software engineering method, to design ethical SIPAs that can reason about context, norms, values, and preferences among values. The preliminary results from our pilot study indicate that priming with values offers significant guidance to participants in making policy decisions. We conjecture that when SIPAs are made aware of such value preferences, they will choose ethical actions and offer a high-quality social experience to the stakeholders. However, these results are based on a small and biased sample without interaction with a production SIPA.

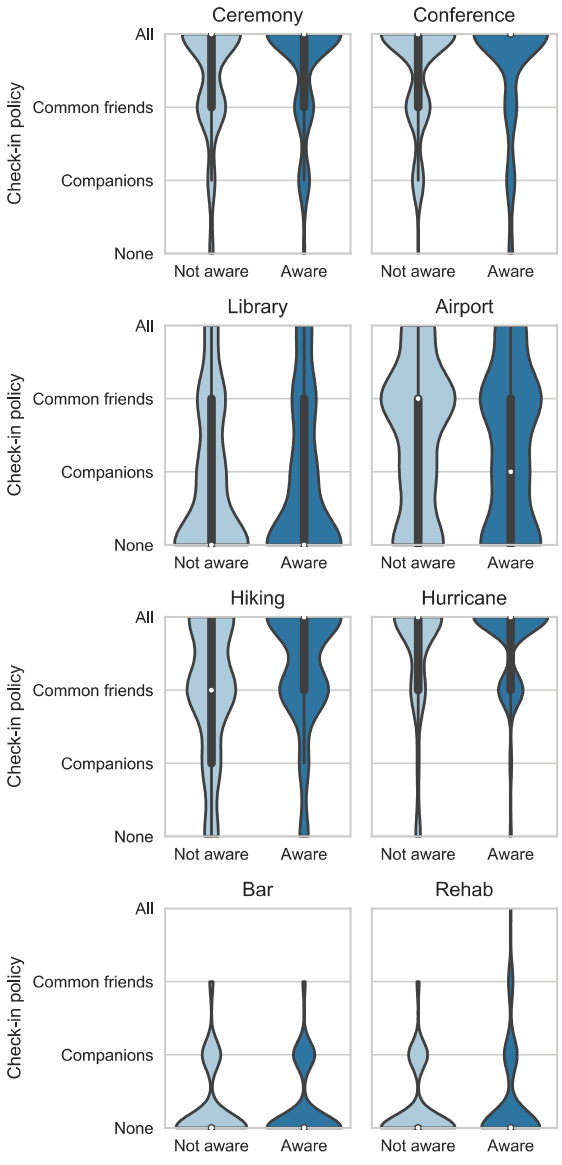


Figure 2. Policy when not aware of values versus when aware of values.

This topic suggests interesting future directions. One, to evaluate the effectiveness of Valar via a developer study. Two, to crowdsource data about values and decision making about sharing policies on a much larger scale. Three, to employ machine learning to assist SIPAs in learning value preferences of stakeholders, and accordingly select policies.

ACKNOWLEDGMENTS

We thank the US Department of Defense for support through the Science of Security Lablet at North Carolina State University.

This article originally appeared in
IEEE Internet Computing, *vol. 22, no. 2, 2018.*

REFERENCES

1. M. Rokeach, *The Nature of Human Values*, Free Press, 1973.
2. P. Pasotti, M.B. van Riemsdijk, and C.M. Jonker, "Representing human habits: Towards a habit support agent," *Proc. 10th International Workshop on Normative Multiagent Systems* (Nor MAS 16), 2016.
3. F. Dechesne et al., "No smoking here: values, norms and culture in multi-agent systems," *Artificial Intelligence and Law*, vol. 21, no. 1, 2013, pp. 79–107.
4. M.P. Singh, "Norms as a basis for governing sociotechnical systems," *ACM Transactions on Intelligent Systems and Technology*, vol. 5, no. 1, 2013, p. 21:1.
5. G.H. Von Wright, "Deontic logic: A personal view," *Ratio Juris*, vol. 12, no. 1, 1999, pp. 26–38.
6. L.G. Nardin et al., "Classifying sanctions and designing a conceptual sanctioning process model for socio-technical systems," *The Knowledge Engineering Review*, vol. 31, no. 2, 2916, pp. 142–166.
7. N. Ajmeri et al., "Arnor: Modeling social intelligence via norms to engineer privacy-aware personal agents," *Proc. 16th International Conf. Autonomous Agents and Multiagent Systems*, 2017, pp. 230–238.
8. P.K. Murukannaiah and M.P. Singh, "Extending Tropos to engineer context-aware personal agents," *Proc. 14th International Conf. Autonomous Agents and MultiAgent Systems*, 2014, pp. 309–316.
9. K. Sotala, "Defining human values for value learners," *Proc. AAAI Workshop on Artificial Intelligence AI, Ethics, and Society*, 2017, pp. 113–123.
10. A. Kayal et al., "Automatic resolution of normative conflicts in supportive technology based on user values," *ACM Transactions on Internet Technology*, 2017.
11. A. Borning and M. Mueller, "Next steps for value sensitive design," *Proc. SIGCHI Conf. Human Factors in Computing Systems*, 2012, pp. 1125–1134.
12. M.O. Riedl and B. Harrison, "Using stories to teach human values to artificial agents," *Proc. AAAI Workshop on Artificial Intelligence AI, Ethics, and Society*, 2016, pp. 105–112.

AUTHOR BIOS

Nirav Ajmeri is a PhD student in Computer Science at North Carolina State University. His research interests include artificial intelligence, multiagent systems, and software engineering with a focus on security and privacy. Ajmeri has a BE in Computer Engineering from Sardar Vallabhbhai Patel Institute of Technology, Gujarat University. Contact him at najmeri@ncsu.edu.

Hui Guo is a PhD student in Computer Science at North Carolina State University. His research interests include multiagent systems, NLP, text mining, and crowdsourcing. Guo has an MS in Computer Science from East Carolina University, and a BS from Tsinghua University. Contact him at hguo5@ncsu.edu.

Pradeep K. Murukannaiah is an Assistant Professor at Rochester Institute of Technology. He received a PhD and an MS in Computer Science from North Carolina State University. Pradeep's research seeks to facilitate the engineering of intelligent personal agents that deliver a personalized, context-aware, privacy-preserving experience to users. Contact him at pkmvse@rit.edu.

Munindar P. Singh is an Alumni Distinguished Graduate Professor in Computer Science and a co-director of the Science of Security Lablet at North Carolina State University. His research interests include the engineering and governance of sociotechnical systems. Singh is an IEEE Fellow, a AAAI fellow, a former Editor-in-Chief of *IEEE Internet Computing*, and the current Editor-in-Chief of ACM *Transactions on Internet Technology*. Contact him at singh@ncsu.edu.

Editor: Bradley Fidler

Do Computers Follow Rules Once Followed by Workers?

Bjorn Westergard

In his 2014 paper "Polanyi's Paradox and the Shape of Employment Growth," economist David Autor puts forward a very general historical thesis (emphasis added):

> *When a computer processes a company's payroll, alphabetizes a list of names, or tabulates the age distribution of residents in each U.S. Census enumeration district, it is "simulating" a work process that would, in a previous era, have been done by humans* ***using nearly identical procedures****. The principle of computer simulation of workplace tasks has not fundamentally changed since the dawn of the computer era. But its cost has. ... This remarkable cost decline creates strong economic incentives for firms to substitute ever-cheaper computing power for relatively expensive human labor, with attendant effects on employers' demand for employees.*

How could a historian of computing adjudicate this claim? How can we determine whether the procedures used by humans and computers are similar, let alone "nearly identical"? Part and parcel with this framing of the issue is Autor's assertion that the inability of workers to articulate the rules they follow when carrying out a task constitutes an impediment to writing software to automate it and his suggestion that this impediment might be overcome with machine learning techniques, which putatively infer these "tacit rules" from a wealth of examples.

Underwriting this view is a theory—henceforth, "the ALM theory"—first laid out by Autor, Levy, and Murnane in *The Skill Content Content of Recent Technological Change* (2003) and *The New Division of Labor* (2004), which builds upon Michael Polanyi's epistemology and attendant conceptions of rule following.

The ALM theory was developed in response to an economic literature that argued that adoption of computer technology—at the level of the industry, firm, or worksite—increases demand for the labor of those with a postsecondary education at the expense of those without. It was thought that in the race between education (supplying computer-complementary skills) and technology (creating demand for them), technology had and would prevail, driving up the wage premia of more educated workers.[1]

This "canonical model" of "skills-biased technical change" employed a binary classification scheme of "more- and less-skilled workers, often operationalized as college- and non-college-educated workers." As the 1990s wore on economists found slowing growth in the college wage premium and nonmonotonic inequality growth difficult to account for in this framework. Subtler distinctions needed to be drawn.[2]

For these, economists pursuing the "task approach" looked to databases of job descriptions, such as the Department of Labor's Dictionary of Occupational Titles and its successor O*NET, to "[measure] the tasks performed in jobs rather than the educational credentials of workers performing those jobs."[3] They would conclude, contrary to the existing skill-biased technical change literature, that beginning in the late 1970s, computerization had issued in "job polarization" or "the simultaneous growth of high-education, high-wage and low-education, low-wages jobs."[4]

The task approach drops the assumption that educational attainment determines work activity in favor of two production functions: one characterizing how labor and computer capital inputs combine to perform tasks, another characterizing how task performances combine to produce outputs (i.e., goods, services). The firm is taken to be a locus of task assignment and execution in which managers play a key role in "organizing tasks into jobs."[5]

The heart of the ALM theory, which is meant to provide an interpretation of the data collected using the "task approach," is the "ALM hypothesis":[6]

> *(1) that computer capital substitutes for workers in carrying out a limited and well-defined set of cognitive and manual activities, those that can be accomplished by following explicit rules (what we term "routine tasks"); and (2) that computer capital complements workers in carrying out problem-solving and complex communication activities ("nonroutine tasks").*

2469-7087/18/$33.00 © 2018 IEEE Published by the IEEE Computer Society

In addition to being "routine" or "nonroutine," tasks are also either "manual" or "cognitive." Example classifications include record keeping, calculation, repetitive customer service (routine cognitive), medical diagnosis, legal writing, managing others (nonroutine cognitive), picking/sorting, repetitive assembly (routine manual), janitorial work, truck driving, and removing paper clips from documents[7] (nonroutine manual).[8]

Routine tasks are "those that can be accomplished by following explicit rules" (equivalently: "precise, well-understood procedures" or "instructions").[9] To count as sufficiently "rules-based"[10] ("procedural," "codifiable") a task must be susceptible to being "fully described in a sequence of if-then-do steps."[11]

Autor found the existing "skills-biased technical change" literature deficient for having failed "to answer the question of what it is that computers do."[12] To fill this lacuna, he suggests that what computers do ("fundamentally") is follow "procedures meticulously laid out by programmers."[13] Routine tasks, then, are "tasks that follow an exhaustive set of rules *and hence* are readily amenable to computerization"[14] (emphasis added).

Nonroutine tasks, by contrast, are those we "only tacitly understand how to perform":[15]

> *But the scope for substitution [of computer capital for labor] is bounded: engineers cannot program a computer to simulate a process that they (or the scientific community at large) do not explicitly understand. This constraint is more binding than one might initially surmise because there are many tasks that we understand tacitly and accomplish effortlessly for which we do not know the explicit "rules" or procedures. I refer to this constraint as Polanyi's paradox, following Michael Polanyi's (1966) observation that, "We know more than we can tell." When we break an egg over the edge of a mixing bowl, identify a distinct species of bird based only on a fleeting glimpse, write a persuasive paragraph, or develop a hypothesis to explain a poorly understood phenomenon, we are engaging in tasks that we only tacitly understand how to perform.*

Autor muses that programming is no longer the only way a machine can come by the rules it follows. Advances in "machine learning" have opened the way to "program[ming] a machine to master the task autonomously by studying successful examples of the task being carried out by others," thereby "inferring the rules that *we tacitly apply but do not explicitly understand*" (emphasis added). The machine running such algorithmically generated code cannot, alas, "'tell' programmers why they do what they do."[16]

The economist Daniel Susskind has recently put forward a critique of the ALM theory on this point.[17] Granting that "a machine must be set an explicit set of programmed rules," he insists that the ALM theorists erred in asserting that "these explicit rules must originate with, and precisely reflect, the thinking process of a human being." Rather than "allowing us to uncover more of the tacit rules that human beings follow in performing 'non-routine' tasks" machine learning systems "allow us to perform tasks with systems and machines that follow rules which do not need to reflect the rules that human beings follow at all, tacit or otherwise." Neural networks that classify skin discolorations, for example, can "derive a set of diagnostic 'rules' that do not need to reflect those that a dermatologist might follow." Therefore, not only can computer capital be expected to substitute for labor in the execution "routine" tasks—those for which we can exhaustively state the rules we follow—but also "routinizable" tasks—which have "features that make it more or less feasible to articulate a set of rules for a machine to follow."

There are two central conceptual issues with both the ALM theory and Susskind's critique thereof. Clarity on these points can help us to avoid historiographic missteps.

First, the pivotal notion of an "exhaustive" set of rules is obscure. In the first half of the ALM hypothesis, the ALM theorists do not merely wish to remind us that computer capital can substitute for labor in carrying out those tasks that we can accomplish by programming a computer (a near tautology). Rather, they wish to explain this susceptibility to "[codification] … in software" of certain work rules (the explanandum) by reference to the "exhaustiveness" of those rules (the explanans).

If the "exhaustiveness" of work rules is simply *defined* as their susceptibility to "codification," there is no distinction between explanans and explanandum, and the ALM theory is explanatorily vacuous (as would be the case if Susskind's "routinizibility" superseded the ALM theorists' "routineness," as he proposes).

The ALM theorists, on the other hand, require but do not provide a criterion of specificity ("exhaustiveness") distinct from the aforementioned susceptibility that is applicable to rules for (e.g.) identifying bird species and multiplying numbers alike. But as sociologist Kjeld Schmidt points out, "the criterion of adequate specification of skilled performance is surely whether the specification serves the purpose for which it is given."[18] Pace Autor, experienced chefs can provide rules specific enough to guide apprentices in cracking eggs ("no, no, strike the flattest part of the egg, like this!"). It is a mistake to think that "even if an account is satisfactory to the practitioners (masters and apprentices alike) for their practical purposes, something can be construed as unsaid."[19] An

engineer writing a program for an egg-cracking robot might define the "flattest" part of the egg in terms of the second derivative of a curve fit to a raster image of the egg, etc.—but this is not a more precise statement of the chef's rule. It is different rule for a different (but related) purpose (writing software to crack eggs) (cf. Shanker on algorithms).[20]

Second, it is unclear what following a tacit rule would entail. The philosopher Stuart Shanker reminds us that in everyday usage "to say of an agent that he/she/it is following a rule, then he/she/it must exhibit the ability to, e.g., instruct, explain, correct, or justify his/her/its behaviour by reference to the expression of the rule."[21] The ALM theorists flout everyday usage. If, for example, we say an individual's ability to identify a bird's species involves the "tacit application" of rules (an application without awareness of a rule the individual cannot express), how can we account for a mistake in identification? The philosopher and social scientist Nigel Pleasants wryly surveys our various (and variously baffling) options: "have they selected the wrong tacit rule; misapplied the correct tacit rule; or followed no tacit rule at all?"[22] By the same token, it is equally meaningless to assert that "machine learning" programs produce rules that resemble (Autor) or do not resemble (Susskind) the rules tacitly followed by workers, because these latter rules are (ex hypothesi) ineffable, and thus unavailable for comparison.

But can't we compare those work rules that can be made "explicit" with the "procedures meticulously laid out by programmers" in "stored instructions (programs)"[23] that "computers follow"[24] and find that they are "nearly identical"? No, because the computing artifacts in question do not follow rules.[25] But their users do, at least some of the time.

In fact, a "new division of labor" did accompany the diffusion of computing artifacts, but it remained a division of labor among human beings. There certainly are algorithms that were applied to tasks both before and after the advent of stored program digital computers, but in these cases the transition was not from human to machine application of a given algorithm, but from application in one set of occupations (e.g., clerk, computer, the managers thereof) to application in another (e.g., engineer, programmer). To demonstrate that the same rules were applied by two such groups, the historian must show that those in the latter appealed to rules expressed by those in the former.

Although this historiographic maxim has not, to my knowledge, been stated before, leading historians of computing already act in accord with it. For example, when Knuth argues that distribution-sorting algorithms "were used [by machine operators] to sort punched cards for many years, long before electronic computers existed"[26] before being "adapted to computer programming," he identifies a paper by computer scientist H.H. Seward[27] (1954) as having been crucial in convincing computer users that "radix sorting within a computer" was feasible. In other words, he justifies his claim that the same rules were applied by machine operators in sorting and computer users in writing sorting programs by providing us with an example of an appeal by a representative of the latter group (Seward) to expressions of rules followed by the former.[28]

References and Notes

1. C.D. Goldin and L. F. Katz, *The Race Between Education and Technology*, Harvard University Press, 2007.
2. J. Schmitt, H. Shierholz, and L. Mishel, "Don't Blame the Robots: Assessing the Job Polarization Explanation of Growing Wage Inequality," Economic Policy Institute, 19 November 2013; http://www.epi.org/publication/technology-inequality-dont-blame-the-robots/.
3. D.H. Autor, F. Levy, and R.J. Murnane, "The Skill Content of Recent Technological Change: An Empirical Exploration," *Quarterly Journal of Economics*, vol. 118, no. 4 (2003), p. 127–1333.
4. D. Autor, "Polanyi's Paradox and the Shape of Employment Growth," vol. 20485, National Bureau of Economic Research, 2014, p. 130.
5. D.H. Autor, F. Levy, and R.J. Murnane, "Upstairs, Downstairs: Computers and Skills on Two Floors of a Large Bank," *ILR Review*, vol. 55, no. 3, 2002, p. 433.
6. D. Susskind, "Re-Thinking the Capabilities of Machines in Economics," Department of Economics Discussion Paper Series, University of Oxford, May 2017; https://www.economics.ox.ac.uk/materials/papers/15127/825-susskind-capabilities-of-machines.pdf.
7. Autor et al., "Upstairs, Downstairs," fn. 16.
8. Autor et al., "Skill Content," p. 1286.
9. Autor et al., "Skill Content," p. 1280.
10. Autor et al., "Upstairs, Downstairs," p. 432.
11. Autor et al., "Upstairs, Downstairs," p. 439.
12. Autor et al., "Skill Content," p. 1280.
13. Autor et al., "Polanyi's Paradox," p. 134.
14. Autor et al., "Polanyi's Paradox," p. 135.
15. Autor et al., "Polanyi's Paradox," p. 135.
16. Autor et al., "Polanyi's Paradox," fn. 38.
17. Susskind, "Re-Thinking the Capabilities."
18. K. Schmidt, "The Trouble with 'Tacit Knowledge'," *Computer Supported Cooperative Work (CSCW)*, vol. 21, no. 2–3, 2012, p. 190.
19. Schmidt, "The Trouble with 'Tacit Knowledge'."
20. S.G. Shanker, "Wittgenstein versus Turing on the Nature of Church's Thesis," *Notre Dame Journal of Formal Logic*, vol. 28, no. 4, 1987, p. 637.
21. Shanker, "Wittgenstein versus Turing on the Nature of Church's Thesis," p. 621.

22. N. Pleasants, "Nothing Is Concealed: De-centring Tacit Knowledge and Rules from Social Theory," *Journal for the Theory of Social Behaviour*, vol. 26, no. 3, 1996, p. 251.
23. Autor et al., "Skill Content," p. 1282.
24. Autor et al., "Polanyi's Paradox," p. 134.
25. Computer programs are sets of rules that can be followed (e.g., by an interviewee solving a programming problem on a whiteboard, or a developer "translating" or "porting" a program), even if computers do not follow them. This is one side of the "duality" of programs, as explicated in R. Turner and N. Angius, "The Philosophy of Computer Science," in *The Stanford Encyclopedia of Philosophy*, spring 2017 edition, Edward N. Zalta (ed.); https://plato.stanford.edu/entries/computer-science/.
26. D.E. Knuth, *The Art of Computer Programming: Sorting and Searching*, vol. 3, Pearson Education, 1998, p. 168.
27. H.H. Seward, "Information Sorting in the Application of Electronic Digital Computers to Business Operations," Technical paper no. RS-232, Digital Computer Laboratory, Massachusetts Institute of Technology; http://bitsavers.trailing-edge.com/pdf/mit/whirlwind/R-series/R-232_Information_Sorting_in_the_Application_of_Electronic_Digital_Computers_to_Business_Operations_May54.pdf.
28. Seward, "Information Sorting in the Application of Electronic Digital Computers to Business Operations." Endnotes 4 and 5 refer the reader to works on punched card procedure in industrial and scientific contexts, respectively.

Björn Westergard is a software engineer in Washington, DC who has worked on electoral campaigning, on-demand delivery, and logistics applications. His research focuses on the specifically capitalist imperatives that have shaped information technologies. Contact him at bjornw@gmail.com.

This article originally appeared in
IEEE Annals of the History of Computing, *vol. 39, no. 4, 2017.*

ADVERTISER INFORMATION

Advertising Personnel

Debbie Sims: Advertising Coordinator
Email: dsims@computer.org
Phone: +1 714 816 2138 | Fax: +1 714 821 4010

Advertising Sales Representatives (display)

Central, Northwest, Southeast, Far East:
Eric Kincaid
Email: e.kincaid@computer.org
Phone: +1 214 673 3742
Fax: +1 888 886 8599

Northeast, Midwest, Europe, Middle East:
David Schissler
Email: d.schissler@computer.org
Phone: +1 508 394 4026
Fax: +1 508 394 1707

Southwest, California:
Mike Hughes
Email: mikehughes@computer.org
Phone: +1 805 529 6790

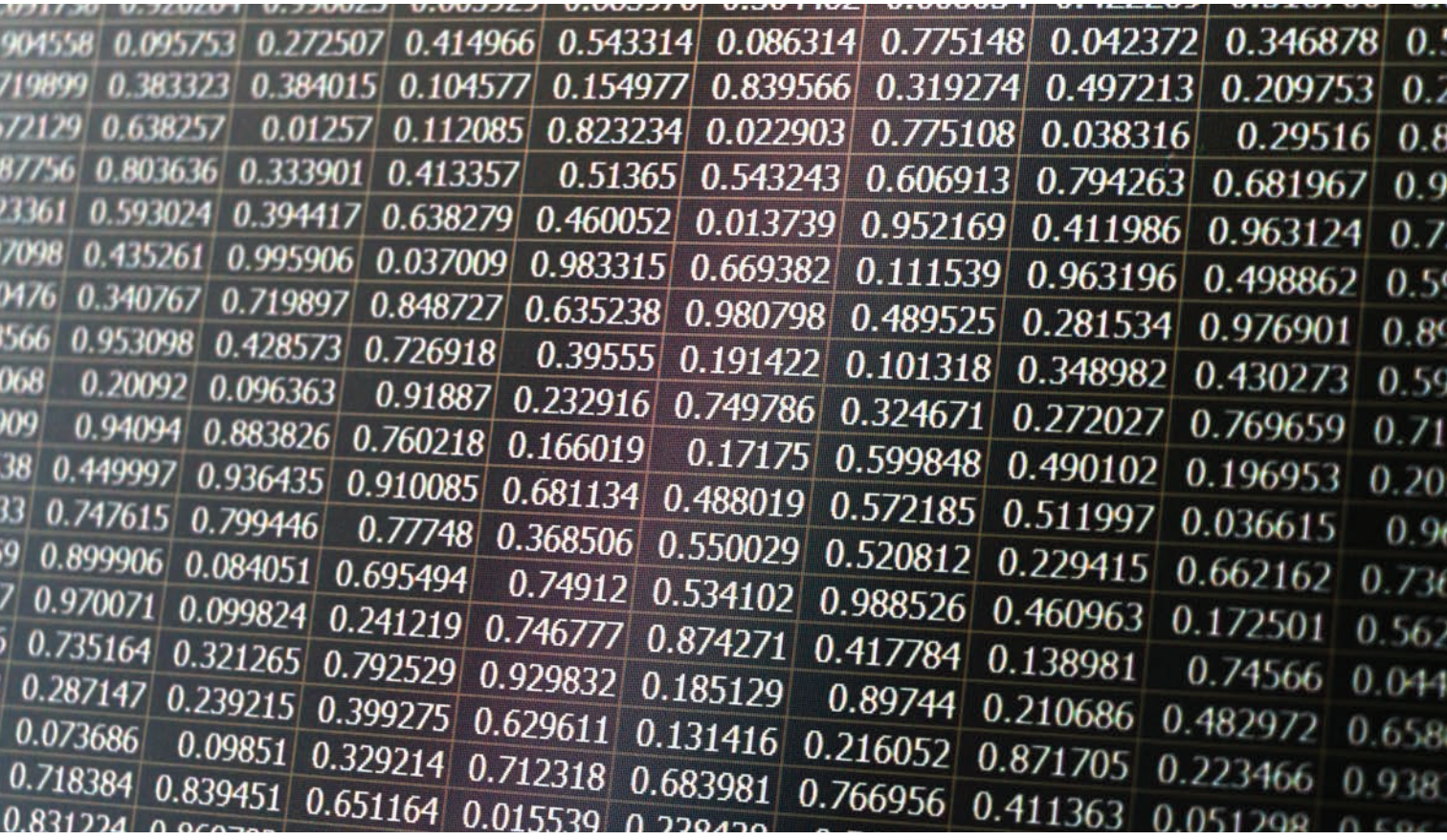
Advertising Sales Representative (Classifieds & Jobs Board)

Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 201 887 1703

Advertising Sales Representative (Jobs Board)

Marie Thompson
Email: marie@4caradio.org
Phone: 714-813-5094

Editors: James J. Hack, jhack@ornl.gov | Michael E. Papka, papka@anl.gov



Large-Scale Calculations for Material Sciences Using Accelerators to Improve Time- and Energy-to-Solution

Markus Eisenbach | Oak Ridge National Laboratory

Understanding the properties of materials is important for a wide range of engineering applications and for answering fundamental questions in condensed-matter physics and materials science. While existing computational capabilities are sufficient to provide both highly accurate results for simple, idealized systems and approximate results for larger systems, the capabilities needed for ab initio investigations of realistic materials, including defects and disorder at operating temperatures, are beyond the practical limits of current systems.

A major and growing impediment to delivering the computational performance needed for next-generation materials science breakthroughs—and similar breakthroughs in any number of other scientific domains—is the amount of power consumed by widely deployed computer architectures. Indeed, power requirements for state-of-the-art leadership computing facilities are now measured in units of megawatts and aren't scalable for many practical reasons. This is one of the major drivers for the introduction of new machine architectures, such as those designed around many-core processors and specialized accelerators.

Toward Modeling Realistic Materials

Although the ground-state properties of a pure compound can readily be calculated with density functional theory (DFT) today, real materials—structures with atomic impurities, crystal

defects, dislocations, grain boundaries, and other low symmetry structures—have to be considered. Yet, the calculation of even the ground state for realistic models of systems of more than 1,000 atoms is a daunting problem for current computer systems.

The need for computational resources at least an order of magnitude more powerful than current machines also arises from the need to calculate the finite temperature properties of these materials. Realistic materials are significantly more complex than idealized materials and need the consideration of a significantly larger number of possible configurations arising from chemical order and atomic displacements, consequently requiring more computational resources. A relevant length scale can be obtained by considering the thickness of a magnetic domain wall, typically a few hundred atomic layers thick.

A computational super-cell to describe the thermodynamics of such a system will require more than 100,000 atoms, which is 100 times larger than the cell sizes used in current-generation computations. Going beyond the static behavior of magnetic systems requires the inclusion of magnetic kinetics in the calculations, driving the need for computational resources capable of calculating realistic systems at finite temperatures well into exascale computing.

One approach to tackling this problem is to use the locally self-consistent multiple scattering (LSMS) method, which was designed to efficiently perform scalable first-principles calculations of materials and condensed-matter systems. The usual approach of finding the solution of the Kohn-Sham equation that underpins DFT relies on expanding the electron wave functions in a way that assumes an ideal, "defectless" periodic crystal with a small cell of representative atoms. This traditional approach scales with the cube of the system size.

An alternative expression for the solution of the Kohn-Sham equation that contains all the necessary information to calculate the physical quantities can be elegantly formulated in the context of the multiple scattering theory for electrons, also known as the Kohn-Korringa-Rostoker (KKR) method. In real space, this allows linear scaling in the number of total atoms in the system for all electron first-principles calculations. Additionally, the main computational requirement of this method lies in inverting dense complex matrices, as the code achieves high compute intensity and benefits from highly optimized dense linear algebra libraries, such as BLAS and LAPACK.

The computational efficiency of LSMS led to its recognition as an outstanding achievement in high-performance computing (HPC), winning the prestigious ACM Gordon Bell Prize in 1998 for enabling the efficient first-principles calculations of ground-state properties of realistic models of materials with disorder or internal nanostructure.

More recently, the capabilities of LSMS have been expanded to investigate the behavior of material properties at finite temperatures. With the extended ability to perform calculations for large simulation cells addressed with the original version of LSMS, calculations of these finite temperature behaviors require the calculation and sampling of a large number of configurations that the atomic sites in these cells can occupy.

This capability was achieved by combining the LSMS code with the Wang-Landau Monte Carlo method to systematically sample randomly selected configurations of simulation cells to compute a system's finite temperature behavior, resulting in the WL-LSMS code. This sampling of configurations introduced an additional level of parallelism that enables the scaling of these calculations to the largest currently available HPC architectures. Thus WL-LSMS was one of the earliest codes to achieve double-precision performance beyond the PFLOP/s mark, which resulted in it being recognized with a Gordon Bell Prize in 2009.

Exploiting the GPU

The large-scale, first-principles simulations enabled with LSMS has allowed a research team using Oak Ridge Leadership Computing Facility's Titan supercomputer to investigate magnetism in large structures and, by using the first-principles-based Monte Carlo simulation, to obtain finite temperature behavior of magnetic materials and alloys. Understanding phase transitions in alloys is of fundamental importance in materials science, and the design of new materials relies on the knowledge of the thermodynamic properties of the different phases. Hence, there's a desire to be able to calculate properties such as phase transition temperatures and specific heat of alloys from first principles.

The advances in available computational resources have made it possible to consider direct simulation of the order-disorder transition in solid-solution alloys without resorting to fitting to models or the need to resort to mean field theories. For example, the team has been able to calculate the ordering transition in brass ($Cu_{0.5}Zn_{0.5}$) without model parameters, where the atoms randomly occupy the lattice sites at high temperature to a regular order at low temperature.

This article originally appeared in
Computing in Science & Engineering, *vol. 19, no. 1, 2017.*

The emergence of hybrid, accelerator-based HPC architectures has opened the door to significant on-node, multithreaded parallelism. To be able to take advantage of these new capabilities of multithreaded and accelerated architectures, the WL-LSMS data layout was restructured to enable bundling multiple atoms into a single MPI process and allow multithreaded execution of multi-atom and multi-energy calculations on a single node.

For the GPU accelerators available on Titan, the team focused on porting the matrix inversion in the multiple scattering part of LSMS to the GPUs, as this part is responsible for roughly 95 percent of all the floating-point operations in the code. The matrix inversion algorithm for general complex matrices employed in LSMS consists of multiple matrix-matrix multiplications and inversions of small submatrices. The matrix-matrix multiplications employ the routines provided in cuBLAS, while the sub-block inversion uses an optimized matrix inversion algorithm that performs LU factorization (a method that transforms a square matrix to two triangular matrices) completely on the GPU device without needing data communication with the CPU host.

To assess the power efficiency of LSMS's GPU port, the team measured Titan's instantaneous power consumption during the execution of the WL-LSMS code. Figure 1 shows the power consumption over time of a simulated cell of 1,024 iron atoms. An identical simulation was performed using a GPU-optimized, CPU-only version of the code. The different phases of the calculation are clearly visible in the power consumption graph, where the high power requirement of LSMS's computationally intensive dense linear algebra calculations are interspersed with the low power demand of the Monte Carlo and communication part of the Wang-Landau calculation. The code's execution pattern is clearly visible in this power trace, both for the CPU and GPU versions. This measurement utilized 18,561 of Titan's nodes, equivalent to 99 percent of the system's capacity. The GPU-accelerated code executes 8.6 times faster than the version of the code that doesn't utilize accelerators, achieving a sustained performance of 14.5 Pflops versus 1.86 Pflops for the nonaccelerated code.

Moreover, the energy consumption for this calculation of the GPU version was 3,500 kilowatt hours, while the identical calculation using only CPUs consumed 25,700 kilowatt hours. Consequently, while the peak instantaneous power consumption of the accelerated version of LSMS is 30 percent higher than the nonaccelerated version, the simultaneous speedup results in a 7.3-fold reduction in energy to solution.

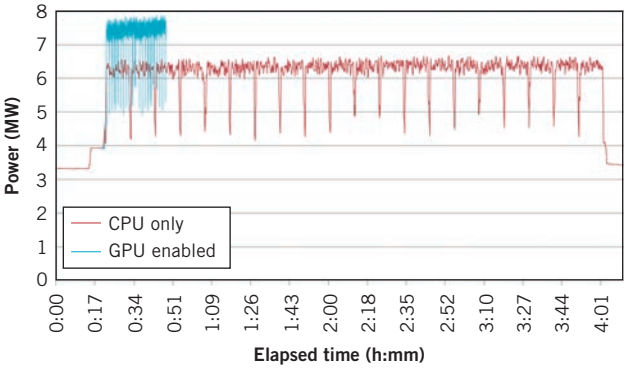


Figure 1. Power consumption of WL-LSMS on identical runs of 1,024 iron atoms on 18,561 nodes of Titan. The power trace shows 20 Monte Carlo steps for each walker. The GPU-enabled version of the code shows significantly increased instantaneous power consumption, with 14.5 Pflops sustained performance for GPU code versus 1.86 Pflops for CPU only. Runtime is 8.6x faster for the accelerated code, and energy consumed is 7.3x less. The GPU-accelerated code consumed 3,500 kWh, and the CPU-only code consumed 25,700 kWh for the same calculation.

At the same time, LSMS maintains scalability and achieves weak scaling efficiency of 96 percent when scaling from a 16-atom calculation on 4 Titan nodes to 65,536 atoms on 16,384 nodes.

These results illustrate the benefits of exploring new, more power-efficient architectures by refactoring algorithms and, in the process, enabling new scientific capabilities. ■

Acknowledgments

This work has been sponsored by the US Department of Energy, Office of Science, Basic Energy Sciences, Material Sciences and Engineering Division (basic theory and applications) and by the Office of Advanced Scientific Computing Research (software optimization and performance measurements). This research used resources of the Oak Ridge Leadership Computing Facility, which is supported by the Office of Science of the US Department of Energy under contract no. DE-AC05-00OR22725. I thank Eric D. Gedenk of Oak Ridge National Laboratory for his help in editing this article. Additional editorial assistance provided by Laura Wolf of Argonne National Laboratory.

Markus Eisenbach is a computational scientist at the National Center for Computational Sciences at Oak Ridge National Laboratory. His expertise lies in first principles calculations for magnetic materials and large-scale scientific computing. Eisenbach received a PhD in theoretical physics from the University of Bristol. Contact him at eisenbachm@ornl.gov.

If You Build It, Will They Come?

Srilatha Manne
Cavium

Bryan Chin
University of California, San Diego

Steven K. Reinhardt
Microsoft

All hardware companies face a conundrum. Should they continue the evolutionary trend of their current products, or build riskier products that have the potential for greater reward but carry a higher probability of failure? The safe course, and one that many customers ask for, is the former. However, as Clayton Christensen points out in *The Innovator's Dilemma*, "most companies with a practiced discipline of listening to their best customers and identifying new products that promise greater profitability and growth are rarely able to build a case for investing in disruptive technologies until it is too late."[1]

Computer hardware companies expend enormous resources to successfully improve their products in an evolutionary fashion. Single-threaded processor performance has been improving at a rate of 15 to 20 percent per year by utilizing both process technology and architectural improvements.[2] These improvements, however, are increasingly difficult to achieve. Using data from Moein Khazraee and colleagues,[3] Figure 1 shows that a processor's cost per operation, as defined by a combination of fabrication, nonrecurring engineering (NRE), and packaging costs, has not significantly improved in the past decade. However, performance improvements are flattening out due to power restrictions and the breakdown of Dennard scaling. For instance, Intel is no longer relying on the tick-tock model, which it rode to market dominance for the past decade, due to the declining benefits of process technology scaling.[4]

Sustaining versus Disruptive Technology

Christensen describes the evolutionary process of improvements using the *sustaining technology S-curve* (see Figure 2). For every successful technology, the performance metric is initially flat during development, rapidly improves for a period of time, and flattens out again when the product and/or technology reaches maturity. Sustaining technologies are dominating the processor industry, and these technologies are reaching a plateau.

Sometimes a disruptive technology with a new S-curve will enter the landscape, as shown in Figure 2. Disruptive technologies do not go head to head with mainstream technologies, but they do have features that a few fringe markets value. Typically, disruptive technologies initially underperform, but then rapidly match and exceed the previous technology. Successful companies not only ride their sustaining S-curves but generate new, disruptive curves to improve performance as the current technology curve flattens out. Microprocessors were once a disruptive technology,[1] and the computing landscape over the past few decades is littered with disruptive technologies, from minicomputers to PCs to smartphones to cloud computing. In all these cases, the disruptive technology yielded worse performance in the near-term when using the same cost function as mainstream technology. However, as Christensen maintains, disruptive technologies eventually redefine how performance is measured.

Recent examples of disruptive technologies in processor architecture include GPUs and Arm servers. GPUs were originally designed for 3D graphics processing, but have made significant inroads first in high-performance computing (HPC) and more recently in machine learning. For applications that are similar to those found in SPECint, GPUs underperform general-purpose processors. However, for targeted HPC applications and machine learning, GPUs are overwhelmingly superior.

Arm processors originally targeted power-constrained embedded domains, but have more recently entered the server market with product offerings from companies such as

 Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

Cavium and Qualcomm that address multicore throughput computing.[5,6] A new S-curve could develop for these specialized throughput-based server products—enabled by highly parallelizable shared-memory applications—just as it did with HPC and machine learning in the GPU market.

It took the GPU market nearly two decades to make headway outside of graphics applications, and the Arm server market has resulted in several failures. Christensen notes that this commonplace in disruptive markets is where "[it] is simply impossible to predict with any useful degree of precision how disruptive products will be used or how large their markets will be." So, how does one innovate in a rapidly changing technology landscape where the underlying cost function is in flux? How does a company keep up with the necessary and expensive evolutionary changes, yet also prepare for and justify expending valuable resources investigating disruptive technologies that are inevitable?

The Case for Agility

Companies and their mainstream customers alike are notoriously bad at predicting what disruptive products will take root in the marketplace. There are many instances of high-profile developments that flopped. For example, it is unlikely you are reading this article on your Apple Newton while listening to music on your Microsoft Zune. Conversely, some disruptive technologies have found success in surprising places such as GPUs. Innovation in a rapidly changing landscape is difficult and prone to failure. Therefore, we posit that architects, rather than trying to predict the future, should pursue agility in order to accelerate innovation while minimizing costs. Hardware companies, architects, and the underlying design methodologies and infrastructure must be nimble enough to deal with disruptive technologies that come from within and outside the current technology landscape. The rest of the article presents some ideas on how this may be accomplished.

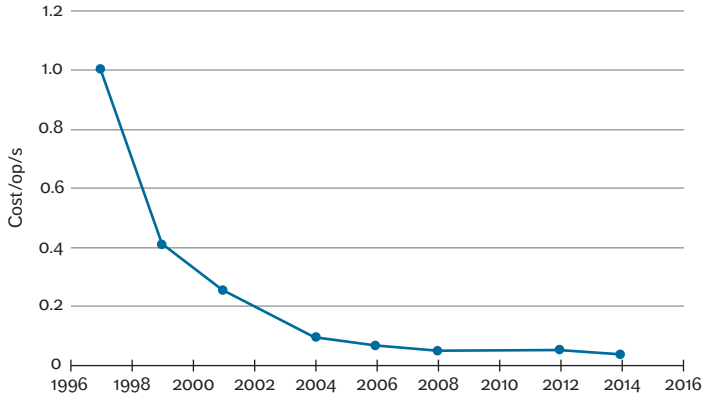


Figure 1. Processor computation cost as a function of time. Cost is defined as a combination of fabrication, nonrecurring engineering (NRE), and packaging costs.[3]

Agile Architecture

In his book *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Business*,[7] Eric Ries writes about software companies that use agile software development strategies. The premise is to deliver prototypes as quickly as possible, even if haphazardly put together, to get early customer feedback. The goal is to use customer feedback to drive product features and direction through a process of continuous development. If you consider how frequently the apps on your phone are updated, or the look and feel of social networking sites evolve, you have seen agile software practices in action.

Facebook, for example, uses agile coding practices. As Kent Beck explains,[8] one of the basic practices at Facebook is reversibility. If a decision is reversible, it does not require the rigorous testing that irreversible decisions require. Code is also released incrementally to a small subset of users, which enables changes to be rolled back with minimal disruption if a problem is found.[9] The challenge for the hardware industry is how to adapt a similar agile methodology without incurring large overheads. We address this challenge in both traditional processor hardware methodologies and innovative methodologies utilized by large computing companies.

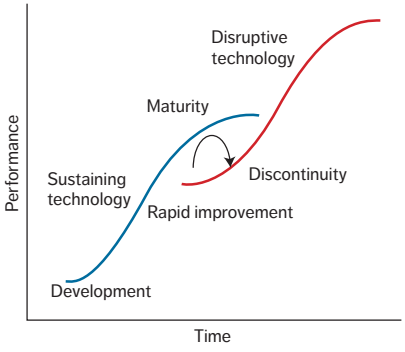


Figure 2. The sustaining technology and disruptive technology S-curves.

Processor Agility

Prior to the ASIC revolution of the past few decades, hardware prototypes were a common means of achieving the rapid development and early feedback cycle. Old technologies such as wire-wrap, breadboards, programmable logic devices (PLDs), and low-cost printed circuit boards (PCBs) enabled hardware companies to quickly build and iterate on products. This methodology is no longer feasible given the complexity and cost of processor development both in terms of engineering time and fabrication costs.[3]

Automated design methodology and reuse. Companies today rely on improved design methodologies and reusability to reduce design time and cost. Design methodologies have made great strides in the past two decades, resulting in shorter design cycle times and an expanded product portfolio using the same fundamental components. Most processors, even those designed for high performance, are mostly or completely synthesized. The Arm roadmap has synthesized cores operating at 3 GHz, and AMD, Intel, and IBM extensively use automated tools throughout their design.[10–12] In addition, companies utilize a modular design methodology such that multiple products can be developed using the same basic components.

Both Intel and AMD use their respective base core designs and innovative packaging technologies to build products ranging from low-power mobile parts to multicore server products.[13] Similarly, silicon companies such as Cavium and Nvidia have been able to create a family of devices with varying price/performance points from the same basic design by utilizing flexible chip layouts that let designers vary the number of computational units and/or the amount of on-die memory. Intel has taken this one step further by collaborating with Facebook to develop a specialized version of Broadwell (referred to as *Broadwell-D*) to meet the specific needs of Facebook.[14]

The technologies mentioned so far reduce design cycle time, but there is still significant overhead associated with bringing a chip to production. Post-silicon functional and performance debug is a formidable challenge for modern processors that may encompass multiple sockets, heterogeneous and/or multithreaded cores, many cores combined with multiple levels of memory hierarchy, complex memory coherence and consistency protocols, and extensive power and performance management via on-chip controllers. In addition, modern processors may operate under complex software stacks containing one or more nested virtual environments. For these reasons, even with mostly synthesized methodologies and reuse of existing components, the transition from first silicon to full production part can take up to a year or more.[15]

Functional verification and bug mitigation. Post-production bugs are commonplace, and fixing bugs in shipped products often involves errata, metal and full-layer spins, and/or replacing existing silicon. Infamous examples of such bugs are the Pentium FDIV bug,[16] the Haswell/Broadwell transactional memory bug,[17] and the AMD TLB bug.[18] These bugs cost the respective companies millions of dollars in lost revenue, and in AMD's case, contributed to its loss of momentum in the server market. All processors have a large list of errata. The table of known errata in Haswell, for instance, covers six pages.[19]

To meet market needs and address the complexity and cost of post-silicon debug, architects must focus on hardware and software solutions for exposing, analyzing, and mitigating functional and performance bugs. Processor vendors must provide tools that rapidly expose and identify bugs and have systems in place for mitigating these bugs without the need for extensive silicon changes. Efforts such as Arm's hardware debug architecture attempt to standardize the infrastructure so that common tools can be made available to the Arm hardware development ecosystem.[20]

Both software and hardware solutions should be explored for mitigating hardware bugs in the field. On the hardware front, microcode fixes on traditional CISC processors come to mind, as does the PAL (Privileged Architecture Library) code feature of DEC's Alpha processors. A similar technology that might help processor vendors mitigate bugs is virtual machine environments. Much software these days is compiled to an abstract machine. Two examples of such abstraction layers, one current and one historical, are Oracle's Java Virtual Machine (JVM)[21] and IBM's AS/400 Series.[22] If an entire processor is designed to execute only a JVM, then the JVM itself provides the instruction set architecture (ISA) of the machine, and the underlying physical machine may have bugs or features that are invisible to the JVM. The JVM addresses ISA-related bugs. Similarly, more fully specified virtual machine environments, such as VMware's vSphere and Microsoft's Hyper-V, virtualize system aspects of the machine, such as memory management and I/O. Machines such as IBM's AS/400 managed to maintain a stable abstract architecture through multiple generations of hardware. By expecting and architecting for bug discovery, analysis, and mitigation, processor vendors can reduce the number of bugs that reach production silicon, and respond to issues in post-production parts quickly and effectively. This shortens the designer-customer feedback loop and leads to a faster development cycle and improved successor products.

Performance verification and optimization. Another critical facet of bringing a processor to production is performance tuning. Processors are designed with dozens of control bits (also referred to as *chicken bits*) to manage system performance. Some chicken bits are exposed to the user (for example, disabling prefetching or simultaneous multithreading mode, or restricting power management), and others are known only by the manufacturer. Regardless, how these bits are set and tuned can have a significant impact on performance. Unfortunately, there are hundreds of these interdependent knobs, and tuning them by hand is impractical. However, self-tuning systems, either integrated into the operating system or as separate tools,[23]

that can dynamically adjust these bits according to application needs may be an innovative mechanism for achieving optimal performance. Best of all, these tuners can be deployed on-site, which means they do not gate product release to customers. Finally, the same techniques for fixing bugs via low-level software or implementing a virtual machine can also be used to adapt silicon to new applications. Hardware designers can enable and deploy new instructions and features through the same mechanisms used to patch around bugs. New versions of a JVM implementation, for example, may exploit optimizations that are relevant to new application areas.

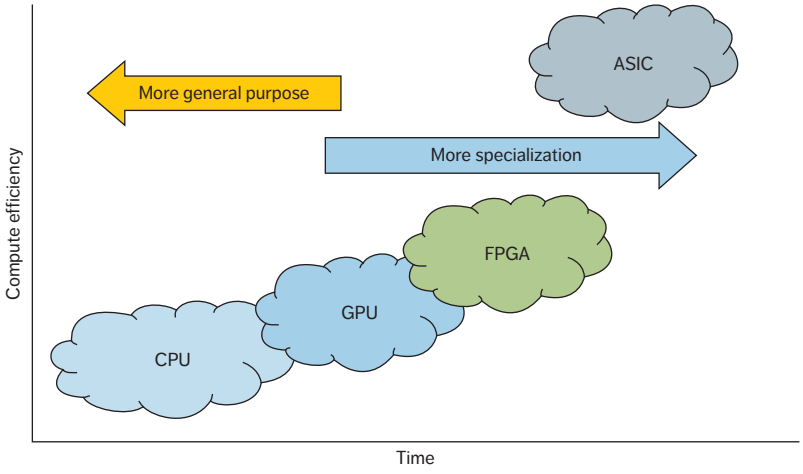


Figure 3. Specialization trend over time.

Computational Agility

So far, we have addressed agility at the processor level. However, with the advent of warehouse-scale systems driven by cloud computing, the processor becomes one piece of a larger computational problem. New companies entering the computing arena include numerous startups and large, established companies from outside the traditional chip design industry, such as Google, Microsoft, and Amazon. Few if any of these companies are choosing to go head-to-head in the general-purpose processor market with traditional designs such as Intel and AMD. Rather, they are achieving agility via specialized devices targeting narrower but highly relevant domains.

The need for specialization. The end of Dennard scaling and the slowdown and imminent demise of Moore's law drive the need for specialization, just as they demand agility in processor design. During the steep part of the S-curve for general-purpose processors, specialized architectures were quickly outpaced by these cheaper commodity devices. The slowing rate of improvement in general-purpose designs both creates opportunity for specialized architectures and drives demand, as customers can no longer rely on the commodity market to satisfy their computing needs.

A prerequisite for specialization is identifying an application or application domain narrow enough to benefit from specialization but large enough to justify a specialized device. Focusing on smaller and smaller domains (down to specific applications) increases the amount of potential performance uplift through specialization, while decreasing the potential market. To be successful, the total value created through specialization (roughly speaking, the value per device times the number of devices) must exceed the cost of developing the specialized device. By developing agile methodologies that reduce engineering costs, we can enable specialization for smaller domains and allow specialized devices to emerge sooner in growing markets.

Figure 3 shows the specialization trend over time, starting with CPUs and ending with custom ASICs. Cryptocurrency mining followed this trend,[24] and deep learning, one of the most prominent new markets attracting specialized architectures, is following suit. GPUs offer better performance than CPUs for certain tasks, such as training for AI, whereas state-of-the art field-programmable gate arrays (FPGAs) can outperform standard GPUs for certain computations such as low-precision arithmetic.[25] Finally, custom ASIC accelerators provide the highest performance efficiency.

Multiple startups such as Graphcore, Wave Computing, Nervana (now part of Intel), and Groq are developing or have developed customized deep learning accelerators that occupy the upper right corner of Figure 3. However, one of the earliest and most publicized deep learning accelerators is not from a startup but from an established company without a history of chip design. The Google Tensor Processing Unit (TPU) was developed in a short 15 months.[26] To achieve a rapid production cycle, Google used an older and more stable process technology (28 nm) and existing communication interfaces. The first-generation TPU was for internal use and had computational and memory bandwidth limitations. However, the TPU is now on its second iteration, and it not only supports higher computational capability and memory bandwidth, but will reportedly be made accessible to third parties.[27]

Even in an agile environment, the delay from the initial ASIC concept

to fully deployed device is measured in years. Once deployed, ASICs must continue to provide value for multiple additional years before replacement. Thus, an ASIC must accelerate a function that, from the point of conception, will still be valuable four to five years in the future. While some functions, such as compression and encryption algorithms, tend to be stable over these time frames, those in rapidly evolving fields such as deep learning may develop new and different requirements in the interval from design start to deployment. Stable, high-volume accelerators can easily justify an ASIC's higher nonrecurring engineering cost. Because an ASIC design needs larger markets and longer lifetimes, an ASIC accelerator typically includes as much flexibility as designers can afford in the form of configuration parameters, options, and software programmability.

To achieve a more agile acceleration framework, Microsoft took an unusual approach to specialization by focusing on FPGAs rather than ASICs for datacenter acceleration.[28] For a given accelerator design, an FPGA implementation could be several times slower and less energy efficient than an ASIC implementation. However, by using hardware devices that can be reprogrammed after deployment, Microsoft gains agility at the expense of computational efficiency. FPGA-based accelerators not only are tolerant to the changing requirements of a given application, but can be completely retargeted as new applications emerge or demand shifts. An FPGA accelerator design can afford to be less configurable and more customized to specific situations, as the design itself can be incrementally modified after initial deployment to address new circumstances. In this fashion, the FPGA's agility as a platform can be used to recover a portion of the efficiency that it sacrifices to an equivalent ASIC-based design.

FPGAs can also close the gap with ASICs by incorporating larger and more complex hard logic blocks on chip. Current FPGAs include multiply-accumulate units and even full microprocessor cores as hard logic. Researchers have also proposed devices that are mostly hard logic, but with configurable interconnect, referred to as *coarse-grained reconfigurable accelerators* (CGRAs).[29] The line between FPGAs and ASICs is further blurred by integrated multichip packages that incorporate both an FPGA and ASIC die.[30] The ability for customers to specify which ASICs are included in the package provides yet another dimension of flexibility.

The computational marketplace. Amazon has also developed hardware for internal consumption from custom routers to chipsets used in its servers.[31] This enables Amazon to optimize the hardware for its specific needs with full control of both the hardware and software stack. Amazon also provides hardware agility to its customers by offering platforms for custom programmable hardware as part of the AWS services plan.[32] The goal is to encourage companies to develop accelerators using Amazon's FPGA framework for internal use and/or sell the resulting computational capability to end customers on the AWS Marketplace. Amazon's EC F1 instances with FPGAs offer two significant benefits for custom solution developers. First, Amazon provides the FPGA hardware, tools, and infrastructure, significantly lowering the cost and convenience threshold for developing customized hardware. Second, Amazon provides a deployment model (via AWS) and a ready marketplace of potential customers for the final product. No longer are hardware developers restricted to products with a large Tier One customer base. They can rapidly develop and deploy niche hardware and test its viability in the AWS computational marketplace with many small customers across the country and the world. The computational marketplace scenario comes closest to achieving the rapid deployment model highlighted in *The Lean Startup*.[7] Finally, if any of these customized solutions become pervasive, they can eventually be reimplemented as an ASIC, as noted by Khazraee,[3] or integrated into a general-purpose processor architecture.

Standardized ecosystem. A successful computational marketplace requires standardized interfaces for interacting with accelerators. On the hardware side, current solutions from Amazon, Microsoft, Google, and others rely on PCIe for accelerator integration. PCIe has been the de facto standard for peripherals for many years, and a part of its success can be attributed to having an open standard. However, for processor designers wanting to create specialized accelerators, PCIe may not offer the tightly coupled memory system integration desired or required by the application. Proprietary coherent processor interconnects such as Intel's QPI and AMD's Infinity Fabric offer the memory system integration that a specialized accelerator might require, while Nvidia's NVLink is a proprietary interconnect for GPUs. Nonproprietary standards from different consortia such as OpenCAPI (www.opencapi.org), Gen-Z (www.genzconsortium.org), and CCIX (www.ccixconsortium.com) might also supplement PCIe as these standards evolve. What is clear, from the PCIe example, is that the new standard should be easily licensable and controlled by an open standards organization to enable a level playing field.

While we have thus far emphasized agility in hardware development and deployment, software agility is also a critical requirement. An environment in which hardware capabilities change and evolve rapidly is impossible to use unless low-level software can adapt equally rapidly, while providing stable

APIs to higher-level services so that the bulk of the code base can remain independent of the underlying implementation's details. Software stacks can provide additional agility when they help to automate the mapping of applications to accelerators, and enable hardware bug workarounds to cope with issues that may slip through an accelerated development and testing schedule.

Processor architecture has changed significantly over the past few decades with the advent of multicore designs, design for low power, heterogeneous systems, and many-core processors that can run a hundred or more threads. With cloud computing and the emerging customizable marketplace of products, we are once again witnessing a sea change in the way computing takes place.

In this article, we have made a case for agility because we cannot predict the future with any level of accuracy. We need agility not only for rapid evolution of conventional architecture, but also for lowering the barrier for specialized architectures. As Bill Gates once noted, "We always overestimate the change that will occur in the next two years and underestimate the change that will occur in the next ten. Don't let yourself be lulled into inaction."[33] As architects, we must develop the infrastructure and mindset that enable us to be agile and take risks in order to evolve with a rapidly changing environment and create the next disruptive technology.

References

1. C.M. Christensen, *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Harvard Business School Press, 1997.
2. "A Look Back at Single-Threaded CPU Performance," blog, 8 Feb. 2012; http://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance.
3. M. Khazraee et al., "Moonwalk: NRE Optimization in ASIC Clouds," *Proc. 22nd Int'l Conf. Architectural Support for Programming Languages and Operating Systems*, 2017, pp. 511–526.
4. J. Hruska, "Intel Formally Kills its Tick-Tock Approach to Processor Development," blog, 23 Mar. 2016; www.extremetech.com/extreme/225353-intel-formally-kills-its-tick-tock-approach-to-processor-development.
5. T.P. Morgan, "Qualcomm Fires ARM Server Salvo, Broadcom Silences Guns," 7 Dec. 2016; www.nextplatform.com/2016/12/07/qualcomm-fires-arm-server-salvo-broadcom-silences-guns.
6. R. Brueckner, "Cavium ThunderX2 Processors Power New Baymax HyperScale Server Platforms," blog, 29 May 2017; http://insidehpc.com/2017/05/cavium-thunderx2-processors-power-new-baymax-hyperscale-server-platforms.
7. E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Business,* Crown Publishing Group, 2011.
8. C. Murphy, "Facebook Guru and Agile Pioneer Kent Beck Reveals the Mind of the Modern Programmer," *Forbes*, 9 Jan. 2017; www.forbes.com/sites/oracle/2017/01/09/facebook-guru-and-agile-pioneer-kent-beck-reveals-the-mind-of-the-modern-programmer.
9. J. Bird, "This Is How Facebook Develops and Deploys Software. Should You Care?" blog, 4 Sept. 2013; http://dzone.com/articles/how-facebook-develops-and.
10. M. Humrick, "Exploring DynamIQ and ARM's New CPUs: Cortex-A75, Cortex-A55," blog, 29 May 2017; www.anandtech.com/show/11441/dynamiq-and-arms-new-cpus-cortex-a75-a55.
11. P. Gelsinger et al., "Such a CAD! Coping with the Complexity of Microprocessor Design at Intel," *IEEE Solid-State Circuits*, vol. 2, no. 3, 2010, pp. 32–43.
12. M. Ziegler, R. Puri, and B. Philhower, "POWER8 Design Methodology Innovations for Improving Productivity and Reducing Power," *Proc. IEEE Custom Integrated Circuits Conf.*, 2014, pp. 1–9.
13. A. Patrizio, "Intel Shakes Up Its Chip Design Process," blog, 23 May 2014; www.itworld.com/article/2699164/hardware/intel-shakes-up-its-chip-design-process.html.
14. V. Rao and E. Smith, "Facebook's New Front-End Server Design Delivers on Performance without Sucking Up Power," blog, 9 Mar. 2016; http://code.facebook.com/posts/1711485769063510/facebook-s-new-front-end-server-design-delivers-on-performance-without-sucking-up-power.
15. M. Abramovici and P. Bradley, "A New Approach to In-System Silicon Validation and Debug," *EE Times*, 16 Sept. 2007; www.eetimes.com/document.asp?doc_id=1276099.
16. "Pentium FDIV Bug," blog; www.cs.earlham.edu/~dusko/cs63/fdiv.html.
17. S. Wasson, "Errata Prompts Intel to Disable TSX in Haswell, Early Broadwell CPUs," blog, 12 Aug. 2014; http://techreport.com/news/26911/errata-prompts-intel-to-disable-tsx-in-haswell-early-broadwell-cpus.
18. K. Kubicki, "Understanding AMD's 'TLB' Processor Bug," blog, 5 Dec. 2007; www.dailytech.com/Understanding++AMDs+TLB+Processor+Bug/article9915.htm.

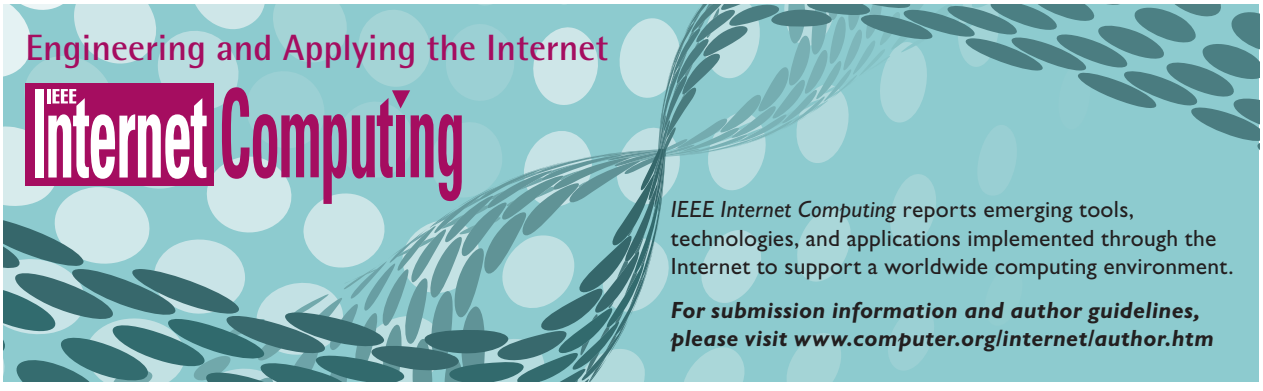
19. *Desktop 4th Generation Intel Core Processor Family, Desktop Intel Pentium Processor Family, and Desktop Intel Celeron Processor Family*, report 328899-037US, Mar. 2017.
20. "Debug Architecture Overview," *ARM*, 2017; http://developer.arm.com/products/architecture/debug-architecture
21. T. Lindholm et al., *The Java Virtual Machine Specification: Java SE 7 Edition*, 28 Feb. 2013.
22. F.G. Soltis, *Inside the AS/400: Featuring the AS/400e Series*, 2nd ed., 29th Street Press, 1997.
23. T. Morad, *The Era of Self-Tuning Servers*, 7 Feb. 2017; www.hpcadvisorycouncil.com/events/2017/stanford-workshop/pdf/Morad_TheEraOfSelfTuningServers.pdf.
24. P. Jama, "The Future of Machine Learning Hardware," blog, 10 Sept. 2016; http://hackernoon.com/the-future-of-machine-learning-hardware-c872a0448be8.
25. L. Barney, "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Learning?" blog, 21 Mar. 2017; www.nextplatform.com/2017/03/21/can-fpgas-beat-gpus-accelerating-next-generation-deep-learning.
26. K. Sato, C. Young, and D. Patterson, "An In-Depth Look at Google's First Tensor Processing Unit (TPU)," blog, 12 May 2017; http://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu.
27. P. Teich, "Under the Hood of Google's TPU2 Machine Learning Clusters," blog, 22 May 2017; www.nextplatform.com/2017/05/22/hood-googles-tpu2-machine-learning-clusters.
28. A. Putnam et al., "A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services," blog, 1 June 2014; www.microsoft.com/en-us/research/publication/a-reconfigurable-fabric-for-accelerating-large-scale-datacenter-services.
29. M. Gao and C. Kozyrakis, "HRL: Efficient and Flexible Reconfigurable Logic for Near-Data Processing," *Proc. IEEE Int'l Symp. High Performance Computer Architecture*, 2016, doi: 10.1109/HPCA.2016.7446059.
30. M. Deo, *Enabling Next-Generation Platforms Using Intel's 3D System-in-Package Technology*, white paper WP-01251-1.5, Intel, Aug. 2017.
31. D. Richman, "Amazon Web Services' Secret Weapon: Its Custom-Made Hardware and Network," blog, 19 Jan. 2017; www.geekwire.com/2017/amazon-web-services-secret-weapon-custom-made-hardware-network.
32. "Amazon EC2 F1 Instances, Customizable FPGAs for Hardware Acceleration Are Now Generally Available," blog, 19 Apr. 2017; http://aws.amazon.com/about-aws/whats-new/2017/04/amazon-ec2-f1-instances-customizable-fpgas-for-hardware-acceleration-are-now-generally-available.
33. B. Gates, *The Road Ahead*, Viking Penguin, 1996.

Srilatha Manne is a principal hardware architect at Cavium. Contact her at bobbiemanne12@gmail.com.

Bryan Chin is a lecturer in the Computer Science and Engineering Department at the University of California, San Diego. Contact him at b5chin@ucsd.edu.

Steven K. Reinhardt is a partner hardware engineering manager at Microsoft. Contact him at stever@microsoft.com.

This article originally appeared in IEEE Micro, *vol. 37, no. 6, 2017.*



TECHNICAL

Oracle America, Inc.

has openings for

TECHNICAL ANALYST-SUPPORT

positions in **Colorado Springs, CO**.

Job duties include: Deliver solutions to the Oracle customer base while serving as an advocate for customer needs.

Apply by e-mailing resume to Sandra.Earnest@oracle.com, referencing 385.19465.

Oracle supports workforce diversity.

TU WIEN FAKULTÄT FÜR !NFORMATIK Faculty of Informatics

Announcement of an open position at the Faculty of Informatics, TU Wien, Austria

FULL PROFESSOR of UBIQUITOUS COMPUTING

The TU Wien (Vienna University of Technology) invites applications for a Full Professor position at the Faculty of Informatics.

The successful candidate will have an outstanding research and teaching record in the field of Ubiquitous Computing and focuses on next generation ubiquitous computing systems and their application in authentic real world settings. Particular research topics of interest include sensor-rich environments; interactive and smart spaces; new interaction paradigms; Internet of Things; mobile and context-aware computing; awareness and privacy; and tangible, situated and embodied interaction.

We offer excellent working conditions in an attractive research environment in a city with an exceptional quality of life.

For a more detailed announcement and information on how to apply, please go to: http://www.informatik.tuwien.ac.at/vacancies

Application Deadline: **October 22, 2018**

TU WIEN FAKULTÄT FÜR !NFORMATIK Faculty of Informatics

Announcement of an open position at the Faculty of Informatics, TU Wien, Austria

FULL PROFESSOR of MACHINE LEARNING

The TU Wien (Vienna University of Technology) invites applications for a Full Professor position at the Faculty of Informatics.

The successful candidate will have an outstanding research and teaching record in the field of Machine Learning, which investigates and develops methods that provide computing systems the ability to automatically learn / improve from experience and to infer or recognize patterns using data, whether for exploratory purposes or to accomplish specific tasks. All areas of machine learning will be considered, covering theoretical foundations, systems, and enabling technologies for machine learning. Methods of interest include, but are not restricted to: statistical machine learning, supervised learning, unsupervised learning, reinforcement learning, deep learning, probabilistic modelling and inference, data analytics and mining, optimization, cognitive systems, neural processing.

We offer excellent working conditions in an attractive research environment in a city with an exceptional quality of life.

For a more detailed announcement and information on how to apply, please go to: http://www.informatik.tuwien.ac.at/vacancies

Application Deadline: **October 1, 2018**

TECHNOLOGY

LinkedIn Corp.

has openings in our **Mtn View, CA** location for:

Software Engineer (All Levels/Types) (SWE0618MV) Design, develop & integrate cutting-edge software technologies.

LinkedIn Corp. has openings in our **Sunnyvale, CA** location for:

Software Engineer (All Levels/Types) (SWE0618SV) Design, develop & integrate cutting-edge software technologies; **Senior Manager, Site Reliability Engineering (6597.58)** Serve as the face of a team responsible for the overall health, performance, & capacity of a significant part of our business; **Staff Engineer, Site Reliability (6597.474)** Ensure that complex, web-scale systems are healthy, monitored, automated, and designed to scale; **Senior Salesforce.com Developer (6597.2449)** Develop, enhance, debug, support, analyze, maintain and test new/existing functionality which supports internal business units or supporting functions; **Senior Front End Developer (6597.2294)** Apply current trends & best practices in front-end architecture, including performance, security, & usability, & own the front-end development for one or more of our internal products.

LinkedIn Corp. has openings in our **San Francisco, CA** location for:

Software Engineer (All Levels/Types) (SWE0618SF) Design, develop & integrate cutting-edge software technologies.

LinkedIn Corp. has openings in our **Carpinteria, CA** location for:

Manager, Database Engineering (6597.1339) Leverage data architecture & warehousing skills to build a leading edge enterprise data warehouse encompassing the entire life cycle, including data integration, transformation, logical & physical design, security, backup, & archival strategies implementing industry best practices.

Please email resume to: 6597@linkedin.com. Must ref. job code above when applying.

Editor: Yong Cui • cuiyong@tsinghua.edu.cn

Autonomic Networking: Architecture Design and Standardization

Xinjian Long, Xiangyang Gong, Xirong Que, and Wendong Wang •
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications

Bing Liu and Sheng Jiang • *Huawei Technologies*

Ning Kong • *China Internet Network Information Center*

Autonomic networking is a promising solution to handle the ever-increasing management complexity of dynamic network environments. This article elaborates on two autonomic networking architectures and current standardization activities revolving around them.

With the ever-expanding scale of the Internet and the advent of new devices and technologies in both wired and wireless environments, the complexity and redundancy of network management are becoming greater challenges to both industry and academia.

Autonomic networking[1] is proposed as a solution to these challenges by introducing the autonomic system engineering into diverse environments (wired, wireless, and so on) and implementing self-managing functions for self-adaptability and context-aware or situation-driven behavior changes in systems, services, or applications. This technology lets us reconfigure the network and optimize it for a nonsupervised or minimal manual administration environment upon the obtained feedback from real-time system behaviors. As a result, the IT system's complexity and maintenance costs are reduced, while dynamically changing user requirements are met.

Autonomic networking is inspired by IBM's autonomic computing.[2] While autonomic computing attempts to improve the closed computing system, autonomic networking addresses the far more heterogeneous and complex network environment. Challenges and difficulties in this area include the following:

- designing an acceptable performant networking architecture to enable the implementation of desirable autonomic behaviors;
- designing novel protocols and exposing exploitable or extendable features from existing protocols to support the handover control, heterogeneity, and cooperation among and within the autonomic networks; and
- designing innovative mechanisms, algorithms, and paradigms to handle different network scenarios such as load balance, information dissemination, and fault detection/removal.

Many projects, such as the European Union's Information Society Technologies (EU-IST) Sixth Framework Program (FP6) Future and Emerging Technologies (FET) exist that design

 Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

and develop clean-slate autonomic elements and architectures, unifying advancements and trends occurring across various areas. These efforts promote continuous improvement and lead the introduction of autonomicity into the generic operational network environment — such approaches include the European Commission's Seventh Framework Program (FP7) project "Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services" (EFIPSANS). Currently, the standardization of autonomic networking is facilitated by IRTF/IETF — through the Network Management Research Group (NMRG) and the Autonomic Networking Integrated Model and Approach (ANIMA), for example — as well as European Telecommunications Standards Institute (ETSI) working groups such as Network Technologies/Autonomic Future Internet (NTECH/AFI) and Next-Generation Protocols Self-X Networks (NGP SXN).

Autonomic networking refers to the act of achieving self-management and adapting to changing environments in accordance with a set of high-level objectives from the network management system rather than performing tasks following static, predefined rules. The most cited properties of self-management are self-configuring, self-optimizing, self-protecting, and self-healing. These properties are realized by the autonomic control loop (ACL), which refers to the procedure of monitoring the managed elements, analyzing the network intelligence, planning policies for self-adaptation, and executing the decided policies in the autonomic network. IBM proposes Monitor, Analyze, Plan, Execute, and Knowledge (MAPE-K) as a reference model for ACL, which also applies to autonomic networking. Currently, novel technologies (such as artificial intelligence) are used in autonomic networking to provide more elastic and reliable network management in mobile networks. Rather than presenting all the existing research efforts, this article focuses on current standardization activities on the networking architecture and its correlated research.

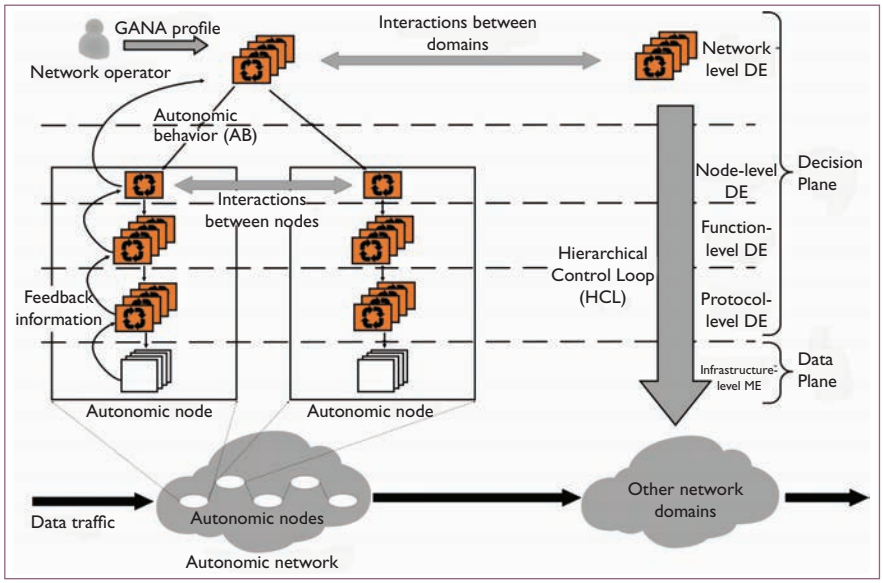


Figure 1. Global view of the Generic Autonomic Network Architecture (GANA) reference model. (DE stands for Decision Element and ME stands for Managed Element.)

Autonomic Networking Architectures

With the advent and rise of autonomic networking, novel networking architectures (such as EU-IST FP6 Autonomic Network Architecture, or ANA) have been proposed. These approaches are considered clean slate and academic in nature. FP7 EFIPSANS targeted IPv6 as the starting point for engineering autonomicity in networks and services. With the Generic ANA (GANA) reference model, standardization communities such as IRTF research group NMRG and IETF working group ANIMA continue to make strides and progress on autonomicity, and these previous works provide vision for ongoing standardization activities.

GANA Reference Model

In EU-IST FP7 Project EFIPSANS, GANA[3] is proposed as a reference model for autonomic network engineering, which implements autonomicity into the Decision Plane and brings a significant reduction in the complexity of network policy computation. Nowadays, GANA is instantiated in various networks by ETSI NTECH/AFI to address self-management in the specific network architecture (for example, the Third-Generation Partnership Project [3GPP][4] and so on). Figure 1 offers a global view of GANA.

GANA adopts (with some modifications) from the Decision, Dissemination, Discovery, and Data Plane network architecture. GANA's Decision Plane is composed of Decision Elements (DEs) and Managed Elements (MEs). According to the scope of decisions, a DE is further classified into four different levels (from the bottom to the top: protocol, function, node, network), and the inferior DEs always serve as the MEs of the superior DEs. This relationship among DEs and MEs in different

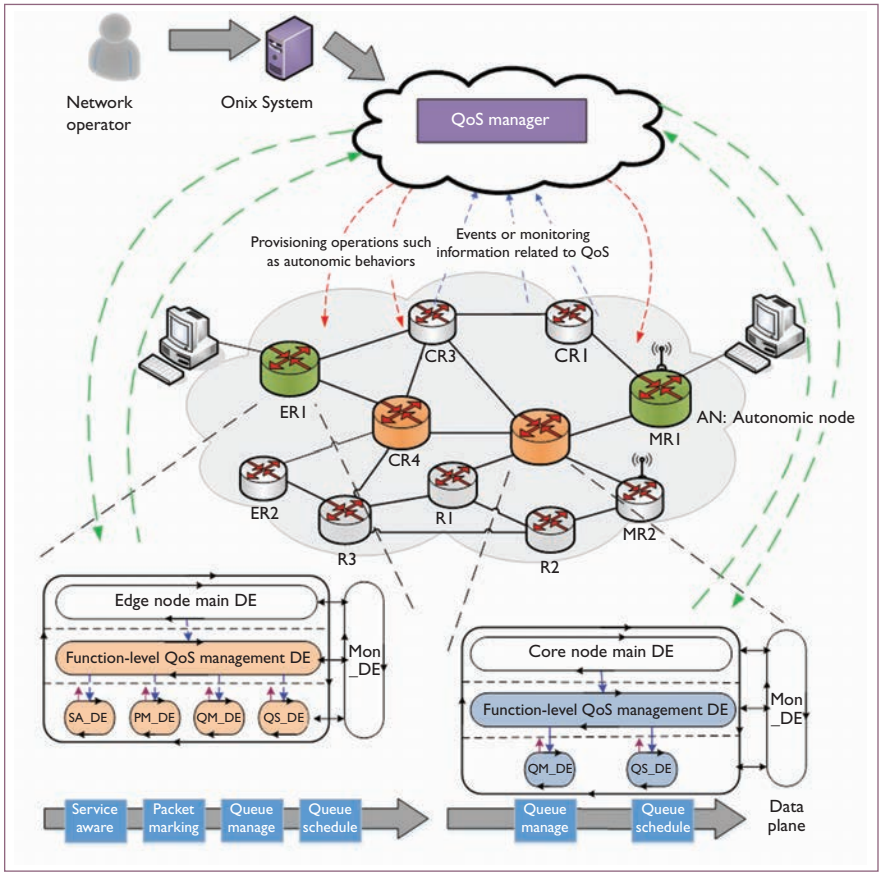


Figure 2. An application scenario of the autonomic architecture for quality of service. (CR stands for core router; ER stands for edge router; MR stands for the router that supports for mobile access; QoS stands for quality of service; and R stands for router.)

levels together forms a Hierarchical Control Loop (HCL) architecture in the GANA's Decision Plane. Policies from superior HCL components are defined as Autonomic Behaviors (ABs).

GANA has been applied in one EFIPSANS' testbed (shown in Figure 2), which improves users' quality-of-service (QoS) experience and reduces manual intervention by automatically adapting to the changing context in IPv6 networks. A set of QoS-related DEs are introduced into both edge and core nodes of the traditional Diffserv architecture. Each DE contains a GANA-based control loop and these control loops cooperate with each other. The network-level DE (known as the QoS manager) distributes ABs to enable the network to self-optimize its resources and self-adapt to the dynamic network context. The node-level DE receives and parses these ABs, then distributes them to the function-level DE. The protocol-level DE follows instructions from the function-level DE and implements specific QoS functions (packet marking, queue management, queue scheduling, and so on).

ANIMA Model

According to RFC 7575[1] and ANIMA's draft,[5] many concepts of the ANIMA model are inspired from GANA. As Figure 3 shows, an autonomic network is composed of autonomic nodes (ANs), and this network might contain more than one autonomic domain. Each AN provides a common set of capabilities across the network called Autonomic Networking Infrastructure (ANI). Autonomic Service Agents (ASAs), which serve as atomic entities of autonomic functions (AFs), are instantiated on ANs. AF refers to the function or the feature that can rely on self-knowledge, discovery, and intent to acquire the information needed for operations without external configuration. ANs and ASAs communicate with each other using a Generic Autonomic Networking Protocol (GRASP) in the Autonomic Control Plane (ACP) created by the ANI. With the help of these communications, AFs run logically over ASAs and span across the network to achieve network-wide autonomicity.

AN consists of three layers: ASAs, ANI (ASA uses services created by ANI), and basic operating system functions. Each AN is assigned a globally unique domain certificate (a logical device identifier, or LDevID),[6] which cryptographically asserts its membership in the autonomic domain, and maintains an adjacency table (containing node-ID, IP address, domain, certificate, and so on) used for recording the ACP neighbor. Each autonomic node maintains a state machine (with three states: factory default, enrolled, and in ACP), which indicates that autonomic networking applies for the whole life cycle of an AN.

ANI is the basis for AFs and is generic to support different ASAs. ANI is composed of three main components: Bootstrapping Remote Secure Key Infrastructures (BRSKI),[7] ACP,[6] and GRASP.[8]

BRSKI. This is an automatic approach to bootstrap a remote secure key infrastructure using vendor-installed X.509 certificates

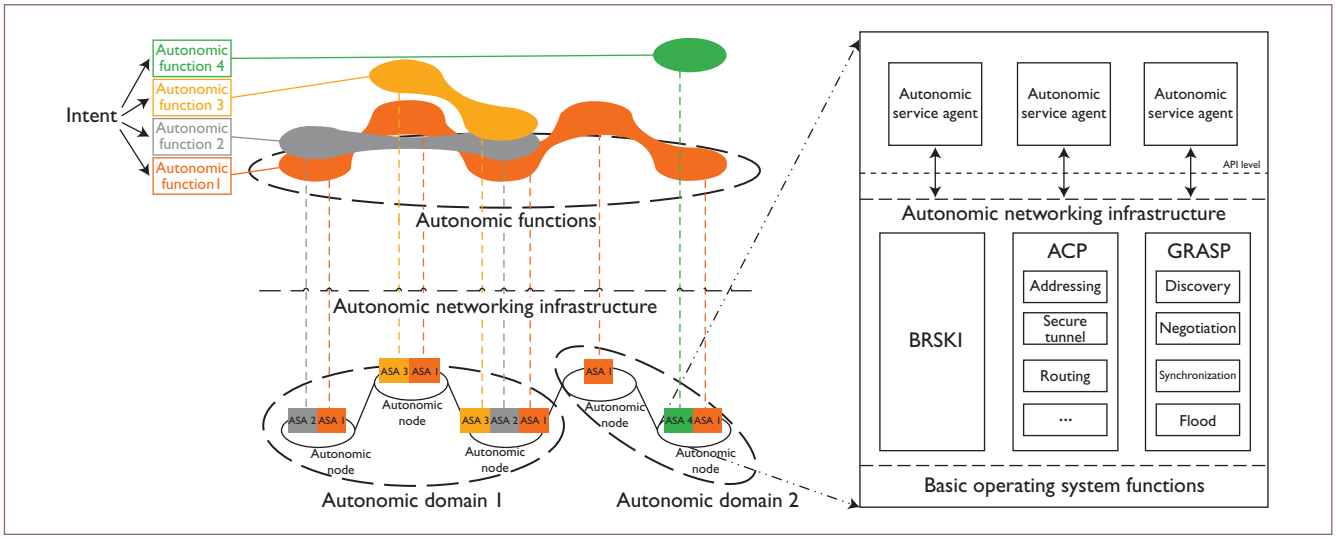


Figure 3. Overview of the (a) Autonomic Networking Architecture (ANA) proposed by Autonomic Networking Integrated Model and Approach (ANIMA), and (b) the structure of an autonomic node. (ASA stands for Autonomic Service Agent; ACP stands for Autonomic Control Plane; BRSKI stands for Bootstrapping Remote Secure Key Infrastructures; and GRASP stands for Generic Autonomic Networking Protocol.)

and a vendor authorized service online or offline. BRSKI secures the initial connection (the distribution of the key materials or the secure certificate in most cases) between the device without configuration (Pledge) and the device that's already in a specific network domain (Registrar). The Pledge actions derive from a cryptographically protected message delivered through the Registrar.

ACP. This is a self-managing and configuration-independent control plane for AF communication. The interaction within the ACP uses IPv6 link local addressing by default. ACP provides management protocols with the functionality of the Virtual-out-of-Band channel, which allows connectivity to all devices regardless of the Data Plane's configuration and the global routing table. ACP also guarantees connectivity for control protocols against the temporal faulty and the transitional event of the Data Plane. Secure tunnels, which are placed into the autonomic node's virtual routing and forwarding instance, are established and create an overlay network after the construction of the ACP. Within the ACP, the loopback interface of each ASA has a routable address using the IPv6 unique local address (ULA) addressing scheme, while the other interfaces exclusively use the IPv6 link local for autonomic functions. Any autonomic device within one ACP has the same /48 prefix. The routing protocol resided in the ACP is independent of the data layer and is mainly used for the distribution of ULA addresses. The ACP's routing protocol is RPL (the Routing Protocol for Low-Power and Lossy Networks, defined in RFC 6550).

GRASP. This protocol will be discussed further in the following chapter.

Standardization Activities on Autonomic Networking

ETSI NTECH/AFI has been working on the application of a GANA reference model onto concrete use cases and setting up an ETSI Specialist Task Force (STF). This STF has proposed two technical reports on using GANA to introduce autonomicity into the 3GPP Core and Backhaul, Ad Hoc, and Mesh network architectures.[4,9] Similarly, IETF ANIMA has been working on further refinement and specification of the main building blocks, signaling protocols, use cases, and other aspects in autonomic networking. In ANIMA's charter, its current target is described as developing one or more protocol specifications.

GRASP

The GRASP generic autonomic signaling protocol for autonomic networking runs in a secure and strongly authenticated communication environment (generally the ACP) by default. Because of a lack of built-in security features, GRASP uses existing mechanisms (such as TLS) to guarantee secure communication in the absence of ACP.

GRASP provides four mechanisms: discovery, negotiation,

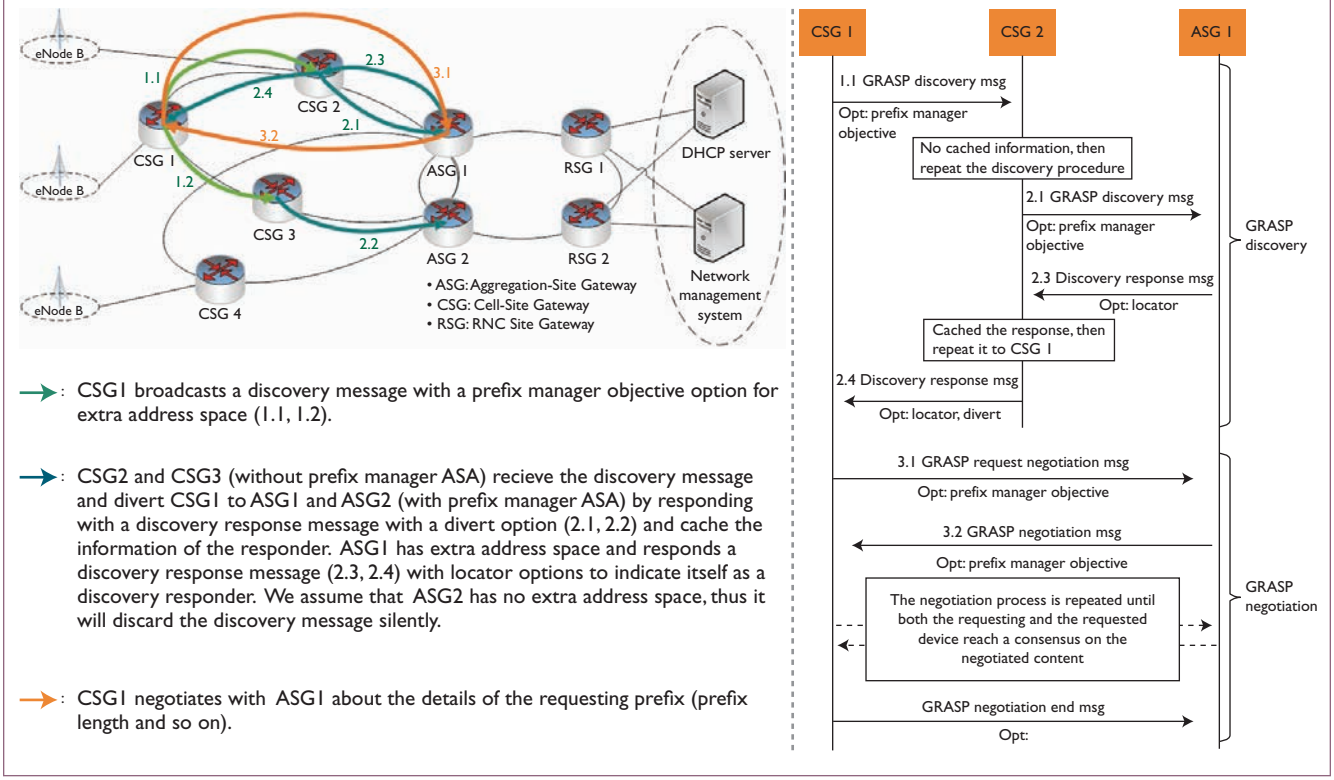


Figure 4. Autonomic IPv6 Prefix Management in Large-scale ISP Networks. The message chart of this scenario is presented on the right. (DHCP stands for Dynamic Host Configuration Protocol.)

synchronization, and flooding. These mechanisms can be combined to provide a rapid mode of operation when necessary.[8] For instance, a session could be ended immediately with an ending message rather than a response message when the counterpart is satisfied with the requested configuration.

Use Case

Features (for example, always-on, data plane independent connectivity, and so on) and main components (such as GRASP) of ANIMA's model apply to various scenarios such as call home, network provisioning, and trouble shooting.

ANIMA proposes a use case (shown in Figure 4) to relieve the human administration of the IPv6 prefix management at the edge of large-scale ISP networks.[10] A C++ implementation developed by the Beijing University of Posts and Telecommunications and a Python implementation developed by the University of Auckland[8,11] are provided as the demo for this scenario.

In the traditional IP Radio Access Network (RAN) solution, a new base station (eNodeB and so on) requests the Cell-Site Gateway (CSG) for a match configuration when it's online. After the request is received, CSG will ask the Aggregation-Site Gateway (ASG) to establish a pseudo wire (PW) for maintenance. Then, eNodeB will request the Dynamic Host Configuration Protocol (DHCP) server for its own IP. CSGs and ASGs (even Radio Network Controller Site Gateways, or RSGs) will serve as the DHCP relay at this stage. The eNodeB will use its IP to communicate with the network management system to obtain all the IPs for services. Similarly, eNodeB will establish a PW for service between the CSG and the ASG. Finally, this new eNodeB comes into service.

In this scenario, prefix management still depends on human planning even with DHCPv6-PD because of the lack of information about the appropriate prefix length that each router should request. In addition, once the PW for service between the CSG and the ASG is established, the endpoints' requirement of timely resource assignments is incompatible to existing protocols. In this case, ANIMA's model is proposed as a solution to achieve the self-configuration — for instance, to enable dynamic IPv6 address space management in large-scale networks.

Compared to the ideal autonomic network, several improvements are needed in the current network, such as more coordination among devices or network partitions; reusable common components; a secure control plane; less configuration; forecasting and closed-loop dry runs for configuration changes; portable network knowledge among network devices; and more efforts on data analysis. As IETF further deepens the standardization, the development of autonomic networking also will be propelled forward. We hope the work and efforts from ETSI, IETF, and others will provide substantial experiences as valuable input to ongoing research on this topic.

Acknowledgments

This work is supported in part by the National High Technology Research and Development Program (863 Program) of China (2015AA016101 and 2015AA015601) and the National Natural Science Foundation of China (61402436)

References

1. M. Behringer et al., *Autonomic Networking: Definitions and Design Goals*, Internet Research Task Force RFC 7575, 2015; www.rfc-editor.org/info/rfc7575.
2. P. Horn, *Autonomic Computing: IBM's Perspective on the State of Information Technology*, IBM TJ Watson Labs, 2001.
3. M. Berringer et al., eds., *A Reference Model for Autonomic Networking*, IETF Internet draft,14 Mar. 2017; https://tools.ietf.org/html/draft-behringer-anima-reference-model-03.
4. M. Berringer et al., eds., *An Autonomic Control Plane*, IETF Internet draft, 28 Mar. 2017; https://datatracker.ietf.org/doc/draft-ietf-anima-autonomic-control-plane.
5. C. Bormann et al., *A Generic Autonomic Signaling Protocol (GRASP)*, IETF Internet draft, 1 Apr. 2017; https://datatracker.ietf.org/doc/draft-ietf-anima-grasp.
6. M. Pritikin et al., *Bootstrapping Remote Secure Key Infrastructures (BRSKI)*, IETF Internet draft, 14 Mar. 2017; https://datatracker.ietf.org/doc/draft-ietf-anima-bootstrapping-keyinfra.
7. S. Jiang et al. *Autonomic IPv6 Edge Prefix Management in Large-Scale Networks*, IETF Internet draft, 11 Mar. 2017; https://datatracker.ietf.org/doc/draft-ietf-anima-prefix-management/?include_text=1.
8. L. Ciavaglia, *Autonomic Network Engineering for the Self-Managing Future Internet (AFI); Generic Autonomic Network Architecture (An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management)*, v. 1.1.1, specification by European Telecommunications Standards Inst., 2013, pp. 80–167; www.etsi.org/deliver/etsi_gs/AFI/001_099/002/01.01.01_60/gs_afi002v010101p.pdf.
9. *Network Technologies (NTECH); Autonomic Network Engineering for the Self-Managing Future Internet (AFI); Autonomicity and Self-Management in the Backhaul and Core Network Parts of the 3GPP Architecture,* tech. report 103 404 (V1.1.1), ETSI, 2016; www.etsi.org/deliver/etsi_tr/103400_103499/103404/01.01.01_60/tr_103404v010101p.pdf.
10. B. Carpenter et al., *Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)*, IETF Internet draft, 17 Feb. 2017; https://datatracker.ietf.org/doc/draft-liu-anima-grasp-api/?include_text=1.
11. *Network Technologies (NTECH); Autonomic Network Engineering for the Self-Managing Future Internet (AFI); Autonomicity and Self-Management in Wireless Ad-Hoc/Mesh Networks: Autonomicity-Enabled Ad-Hoc and Mesh Network Architectures*, tech. report 103 495 (V1.1.1), ETSI, 2017; www.etsi.org/deliver/etsi_tr/103400_103499/103495/01.01.01_60/tr_103495v010101p.pdf.

Xinjian Long is a PhD candidate in the State Key Laboratory of Networking and Switching Technology (SKL-NST) at the Beijing University of Posts and Telecommunications (BUPT). His research interests include artificial intelligence in networks and software-defined networking technologies. Long has a BS in telecommunications engineering with management from BUPT. Contact him at barbiel_origin@bupt.edu.cn.

Xiangyang Gong is a full professor in the SKL-NST at BUPT. His research interests include IP quality of service (QoS), video communications, novel network architectures, and mobile Internet. Gong has a PhD in telecommunication and information system from BUPT. Contact him at xygong@bupt.edu.cn.

Xirong Que is an associate professor in the SKL-NST at BUPT. Her research interests include innovation application, next-generation network architectures, and mobile Internet. Que has an ME in computer applications from BUPT. Contact her at rongqx@bupt.edu.cn.

Wendong Wang is a full professor in the SKL-NST at BUPT. His research interests include future Internet architectures, software-defined networking technologies, and network QoS. Wang has an ME in computer science from BUPT. He's the corresponding author of this article. Contact him at wdwang@bupt.edu.cn.

Bing Liu is a research engineer at Huawei Technologies. His research interests include autonomic networking. Liu has a PhD in signal and information processing from BUPT. Contact him at leo.liubing@huawei.com.

Sheng Jiang is a senior principle engineer at Huawei Technologies. His research interests include autonomic networking and artificial intelligence in networks. Jiang has a PhD in computer science from University College London. Contact him at jiangsheng@huawei.com.

Ning Kong is a researcher at the China Internet Network Information Center. His research interests include Internet naming and addressing technologies as well as the Internet of Things. Kong has a PhD in computer software and theory from the University of Chinese Academy of Sciences. Contact him at nkong@cnnic.cn.

Touradj Ebrahimi
Ecole Polytechnique Fédérale de Lausanne

JPEG at 25: Still Going Strong

Graham Hudson, Alain Léger, Birger Niss, and István Sebestyén
Original JPEG Development Team Members

Photographic images are taken for granted today as part of the multimedia mix of information we use daily to communicate in both our personal and professional lives. An enabling technology for this is the photographic coding technique, universally known as JPEG, which is celebrating its 25th anniversary of receiving approval as standard this year. Where did JPEG come from, and what are the fundamental components that have given it longevity?

The Origins

The image compression technique used for applications as diverse as photography, webpages, medical imaging, and public records is JPEG, named after the original International Standards Organization (ISO)/International Telegraph and Telephone Consultative Committee (CCITT) Joint Photographic Experts Group, established in November of 1986. The group developed the technique in the late 1980s and produced the international standard, formally known as Int'l Telecommunication Union (ITU)-T T.81, in the early 1990s.[1]

How It All Began

In the early 1980s, some of the leading telecommunication service providers around the world were launching videotex services: information services delivered over analog telephone lines to a terminal based on a television set or to an inexpensive dedicated terminal (see Figure 1). Such services were also delivered to PCs once they became available. The technology was primitive by today's standards. The data rates available using a modem over a copper telephone pair were generally 1,200 (download)/75 (upload) bits/s, although up to 4,800 bits/s was possible.

Later in the decade, the 64 kbits/s integrated services digital network (ISDN) was introduced, and many assumed it would eventually become widely available. At that time, Cathode Ray Tube displays were capable of resolutions up to 640 × 400 pixels, and advanced televisions had text and graphics display controllers for teletext. Some even had microprocessors, but RAM was expensive and was typically limited to a few kilobytes. Following the Commodore Pet and Apple 2 computers, the IBM PC was launched. These early machines had displays supporting graphics and color, but they didn't have photographic display capability.

With the advent of ISDN, telecommunication companies' research centers looked for ways to improve their videotex service offerings by enhancing display capabilities, using computer (geometric) graphic and photographic image coding techniques. However, photographic images contain a lot of information. The ITU-R Digital Studio Television Picture Standard recommendation[2] was taken as reference for this early work on picture coding. A full-frame 601 picture has 720 × 575 pixels, as illustrated in Figure 2. It uses a color encoding system known as YCbCr 4:2:2. The chrominance components, Cr and Cb, have half the resolution of the luminance component Y. All components are represented by 8 bits, giving an average of 16 bits per pixel, requiring 828 kbytes per frame for storage and transmission. Even at the ISDN rate of 64 kbits/s, it would take more than 104 s to transmit a full frame.

A great impetus to the international development and evaluation of picture coding techniques was the formation of the European Strategic Program for Research in Information Technology (ESPRIT) project 563—Photovideotex Image Compression Algorithms (PICA) in 1985 (Table 1 lists this and other historical milestones).[3,4] The consortium of seven partners had experience in telecommunications, broadcasting, and computing (see Table 2). The core team included picture coding experts from leading European telecommunication laboratories already involved in international standards activity (see Figure 3). Throughout the PICA

 Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

project's lifetime (1985–1988), key contributions were made to the JPEG technical kernel and to application requirements and the evaluation of coding techniques. In particular, the project developed and evaluated 10 techniques, and two were submitted for standardization (and one of those two was the central part of the future JPEG standard).

Evolving International Standards

Videotex standards were being worked on by international standards bodies responsible for text, graphic, and image coding at CCITT, CEPT (Conférence Européenne des Administrations des Postes et Télécommunications), and ISO. Digital image coding work had typically focused on facsimile, slow-scan television, and teleconferencing. The early work on photographic coding initiated by three European telecommunication labs—British Telecom Labs, CSELT (Centro Studi e Laboratori Telecomunicazioni), and CCETT (Centre Commun d'études de Télévision et Télécommunications)—was presented to the international standards bodies, first in 1982 at CEPT, and later to ISO and CCITT.

In 1982, ISO TC97/SC2 established Working Group 8, chaired by Zak Muscati (Department of Communications of Canada) and later by Hiroshi Yasuda (Nippon Telegraph and Telephone, Japan). The working group was established to define the principles of graphic and photographic coding. Early in 1986, during a meeting in Boston, the scope and progress of the ESPRIT project was presented to this group.

In CCITT Study Group VIII (SGVIII), a special rapporteur's group was formed in 1985 to investigate new forms of image communication. The group was initially chaired by Manfred Worlitzer (Deutsche Bundespost, Germany). Then, in 1987, István Sebestyén (Siemens, Germany) took over. The group analyzed the different coding types (text, graphic, geometric, incremental, and photographic) required for different telecommunication services (facsimile, teletext, videotex, and teleconferencing), and they formulated requirements for common components for image communications.

Realizing the importance of picture coding for future multimedia communication services, in July 1986, the leaders of the CCITT and ISO groups proposed that the ISO Photographic Expert Group (PEG) should become a joint working group (JPEG) to select a high-performance photographic image compression technique, with CCITT setting the service requirements and ISO providing the coding expertise. The first meeting of JPEG under the chairmanship of Graham Hudson (British Telecom, UK) was in November 1986 in Parsippany, New Jersey. Following the agreement of a technique for standardization in 1988, Greg Wallace (Digital Equipment Corporation, US) took over the chair until the JPEG standard was approved first by the ITU and later by the Joint Technical Committee 1 of ISO and IEC.

(a) **(b)**

Figure 1. Information services delivered over an analog telephone line to an inexpensive dedicated terminal: (a) the Minitel terminal in France (source: CCETT; used with permission) and (b) a photovideotex page on the Prestel terminal in the UK (source British Telecom; used with permission).

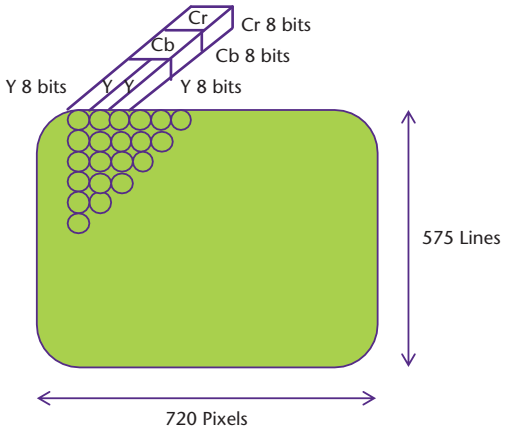


Figure 2. The Digital Studio Television Standard. It uses 8 bits for the luminance component but half the resolution for the two 8-bit chrominance components, giving an average of 16 bits per pixel, requiring 828 kbytes per frame to store and transmit data.

The Wish List

The JPEG working group aimed to find and standardize a compression technique that could be used for a broad range of continuous tone images for applications, ranging from

Table 1. Historical milestones.

Date/location	Milestones
1982	Introduce image coding for videotex at CEPT (Conférence Européenne des Administrations des Postes et Télécommunications)
June 1985, Ipswich	Launch the European Photovideotex Image Compression Algorithms (PICA) project
November 1986, Parsippany	ISO and CCITT form Joint Photographic Experts Group (JPEG)
March 1987, Darmstadt	Register coding schemes and define requirements and selection process
June 1987, Copenhagen	Hold initial selection meeting—10 techniques reduced to 3
October 1987, Washington	Revise specification and hold first selection process
December 1987, Winchester	Revise specification and hold second (final) selection process
January 1988, Copenhagen	Hold final selection meeting—adaptive cosine transform (ADCT) technique chosen
June 1989, Rennes	Refine and consolidate the ADCT technique by the JPEG international team
1989	Write the JPEG draft international standard with ITU/ISO/IEC common template
1992	Approve JPEG as Recommendation ITU-T T.81
1993	Approve JPEG as ISO/IEC 10918-1 Standard

Table 2. The seven partners of the European Strategic Program for Research in Information Technology (ESPRIT) project 563—Photovideotex Image Compression Algorithms (PICA).

Company	Country
BT Labs	UK
IBA	UK
KTAS	Denmark
DNL	Netherlands
CSELT	Italy
CCETT—FT Labs	France
Nixdorf	Germany

photovideotex (the web had not yet been invented) to press photos and medical images.[5] They set about defining a set of mandatory requirements. An essential feature was the ability to adjust the compression factor (the reduction in data) versus the final quality to match the needs of the application.

With restricted data-rate transmission channels, the group considered it mandatory to provide a progressive picture build-up.[6] The idea was to quickly deliver a crude (lower resolution/quality) image for instant display, which could subsequently be improved in several stages until the highest quality was achieved. This facility also provides pictures of different resolution and accuracy to be held on a database and delivered to match the capability of the output device. Sequential build-up, where a full-quality image is built up, from top to bottom and line by line, was also needed for rapid picture file transfer.

The JPEG working group also realized that some applications, such as medical imaging and document archiving, required a final image to be identical to the original. This is referred to as lossless, reversible, or exact coding.

Many of the applications (such as audiographic conferencing and remote screen sharing) presenting an image to a display required the decompression in real time. This was a challenge with the technology of the time (that is, with IBM's PC Advanced Technology (AT) machine, with a 20MHz 386 processor or dedicated digital signal processors[7]), so the three algorithms in competition after the January 1987 Copenhagen meeting were required to show demonstrations of the real-time decoding.

Choosing the Algorithm

The JPEG working group set out to define a procedure to select a coding technique. For a technique to be considered as a candidate for standardization, the proposer had to provide a full technical description and a set of agreed test pictures encoded/decoded at different compression factors. At the first JPEG meeting (Parsippany, Nov. 1986), 14 different techniques were presented, but only 12 proposals were officially registered in Darmstadt in March 1987, at the second JPEG meeting. The candidates included examples of most compression techniques known to the scientific community at the time, such as predictive coding, block coding,

cosine transform, vector quantization, and combinations of these. The key requirements for candidates for the final selection process were as follows:

- provide and present full documentation of the technique including functionality, principles, and implementation;
- prepare and conduct subjective testing of new nonstandard test images for bit rates (0.08 bit/pixel, 0.25 bit/pixel, 0.75 bit/pixel, and 2.25 bit/pixel);
- demonstrate a prototype 64 kbits/s real-time decoder with progressive build-up; and
- submit executable code.

Three techniques stood out at the initial selection process at KTAS in Copenhagen in June 1987—the European (PICA) adaptive cosine transform (ADCT) technique, the US (IBM) differential pulse code modulation (DPCM)-based technique, and the Japanese block truncation coding scheme. These three techniques were used as the basis for further development by international teams led by Europe (Alain Léger), US (Joan Mitchell), and Japan (Yasuhiro Yamazaki) respectively, for the final selection meeting held at the Copenhagen Telecom Company (KTAS) Laboratories in January 1988.

For the final selection, the test requirements were increased. Subjective testing took place at 2.25 bits per pixel, 0.75 bpp, 0.25 bpp, and 0.08 bpp using five new test images for which the candidate algorithms were not trained (see Figure 4). A double stimulus technique was employed, whereby images were compared with the original.[8]

It was evident from the subjective testing (see Figure 5) that the ADCT technique produced higher-quality results for all of the compression stages. Excellent results were achieved at 0.75 bpp (20:1 compression) and results indistinguishable from the original were produced at 2.25 bpp. ADCT real-time decompression was demonstrated in software on an IBM PC AT with a 20MHz 386 processor. The group unanimously agreed to develop a standard based on this ADCT technique.[9,10]

Key Technical Choices

Here we explain the key technical decisions the JPEG group made during the building process of the JPEG compression scheme and format.

Figure 3. The JPEG core team included picture coding experts from all over the world, including leading worldwide telecommunication and IT laboratories already involved in international standards activity.

The Transform

The scientific literature shows the optimum transform is the Karhunen-Loeve Transform (KLT). The KLT analyzes the image and extracts the principle components, thus compacting the energy very efficiently. However, it's computationally intensive—far more than realistically achievable in the late 1980s. Furthermore, the calculated transformation kernel depends on the image content, so it must be calculated for each image.

Various other simpler transforms were examined during the development of JPEG: high and low correlation transforms, where all operations can be done using only shifts and adds, and the discrete cosine transform (DCT), which can be calculated using very fast algorithms (like the Fast Fourier Transform). DCT was by far the best of the second-best options, with an energy compaction approaching the KLT. It was therefore decided to continue with DCT as the transform of choice.

Discrete wavelet transform (DWT) appeared later (with the orthogonal version appearing in 1987), which avoids blocking artifacts, but it wasn't feasible with the hardware of the day and with the speed requirements (real-time decoding at ISDN 64 kilobits per second). JPEG2000 was standardized later (after 2000) with DWT, but it was never intended to replace ADCT in JPEG (1992).

Block Size

From an energy compaction point of view, the optimum block size should be one where the pixels in an average block are correlated. Using too small a block size misses important pixel-to-pixel correlation. Using too large a block size

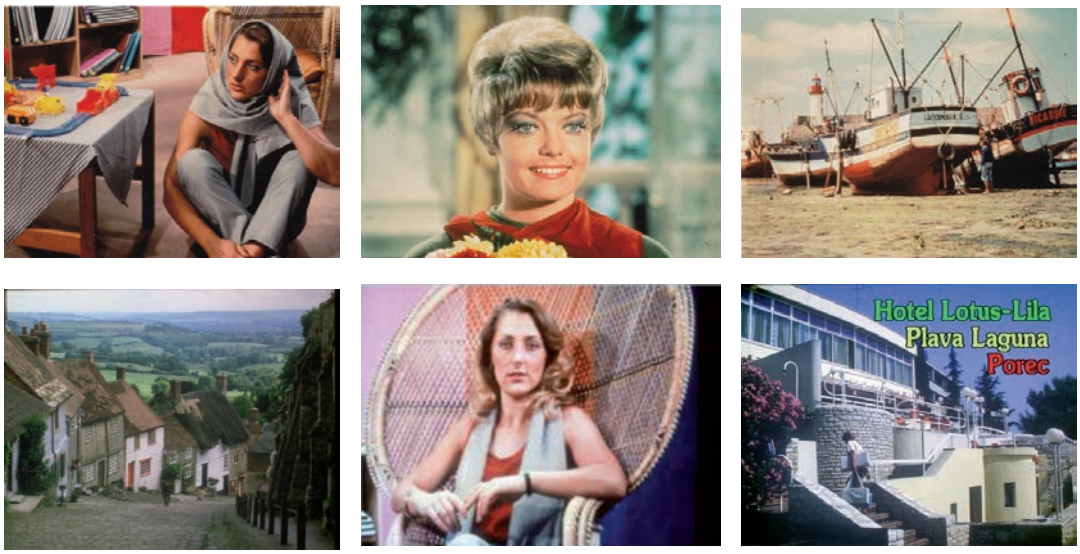


Figure 4. Some JPEG test pictures (source: JPEG; used with permission). These are examples of images used for the first and final selection process for the coding technique. (Note that these are the reproduced images. Credits for the original versions are (from left to right, top row) IBA, SMPTE, and CCETT; (bottom row) Roy Vivian, EBU, and Roy Vivian/IBA).

tries to take advantage of a correlation that might not exist.

Working with the typical image sizes of the late 1980s (720 × 575 pixels), 4 × 4 blocks were too small to catch important correlations, and 16 × 16 blocks often contained uncorrelated pixels and increased calculation complexity for no gain. So out came the 8 × 8 block!

Today, with 4K and 8K and higher display resolutions, larger block sizes (16 × 16 or even higher) are an obvious consideration.

Psychovisual Quantization

Having performed the discrete cosine transform on an 8 × 8 block, 64 pixel values have been transformed into 64 amplitudes of 2D cosine functions of various frequencies. The eye, however, is not equally sensitive to all frequencies. Low-frequency variation within the 8 × 8 block is much more visible than high-frequency variation. This is where quantization comes into play: low frequencies are represented with higher accuracy than high frequencies without jeopardizing the visual content of the blocks.[11] This is generally what provides lossy compression.

During the development of JPEG, researchers considered (and experimented with) the quantization of the less visible dark areas. Blocks with low DC values (dark blocks) could be quantized more harshly than blocks with medium or high DC values. Experiments showed, however, a prominent problem with such content-dependent strategies: adjacent blocks treated with different quantization matrices are visually different and thus add heavily to the annoying blocking artifacts that are seen at high compression without really improving the compression rate. In JPEG, all blocks in a given channel are quantized with the same quantization values.

Modeling and Encoding

Transformation and quantization together produce datasets with a statistical structure that lends itself to complementary compression. The process to ensure this is the modeling (optimal-source symbols selection) and encoding of the selected symbols. Given that the majority of the quantized amplitudes are either zero or very small, and that most of the nonzero or larger quantized amplitudes pertain to the low frequencies, KTAS (primarily Jørgen Vaaben) devised an ingenious way to encode these using value pairs. The first value in the pair tells how many zero amplitudes to skip before the next nonzero amplitude (run length), and the second value in the pair tells how many bits are necessary to represent that amplitude. The value pair is then followed by the amplitude. When there are no more nonzero amplitudes in the block, an end-of-block code is emitted.

The statistical distribution of these value pairs is heavily skewed toward small values of both runs and number of bits, so the 2D

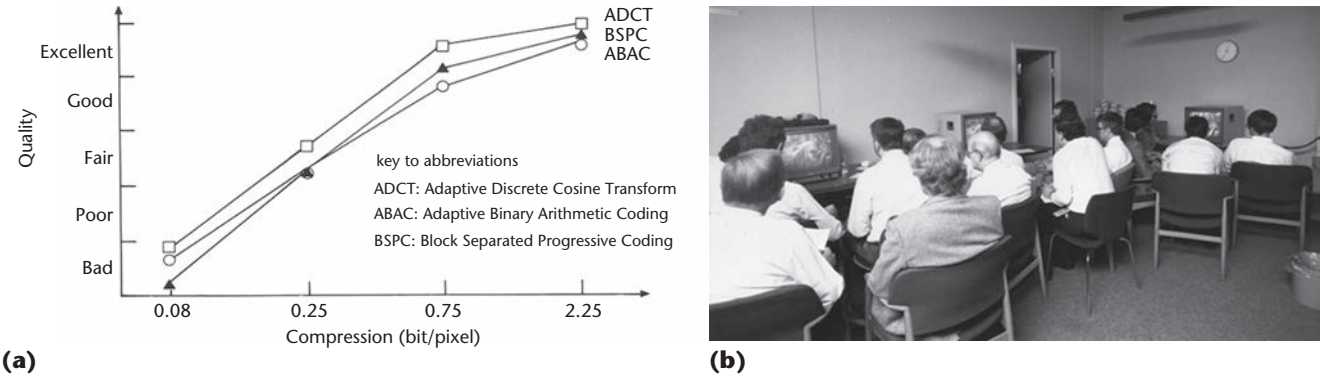


Figure 5. Subjective testing: (a) results from the test, conducted at (b) KTAS in Copenhagen (source Joint Photographic Experts Group; used with permission).

Huffman coding was the obvious choice. With this encoding scheme (lossless entropy coding), significantly higher compression rates were obtained in JPEG.

Baseline and Profiles

In the early stages of drafting the standard (1988), the group proposed producing a kernel that fulfills most of the expected requirements of videotex and envisioned image telecommunication services. The results of the final selection formed the basic kernel (baseline) JPEG system. Most significantly, a royalty-free baseline system was created. On this foundation, other profiles were added like layers of an onion for specific applications and for options such as arithmetic coding. Such options might have been royalty bearing. The baseline coding scheme structure is robust and has a very low algorithmic complexity, making it easy to understand and implement. The baseline is sufficient for the many applications and is heavily used.

This patent strategy for the JPEG baseline and options proved to be most successful in supporting market penetration of the JPEG algorithm. On this basis, the Independent JPEG Group (an informal open source group under the leadership of Tom Lane) released an open source JPEG code in October 1991 (based on the draft JPEG standard). At that time, the Internet and the web badly needed a still-picture compression standard.

Later (2000–2002), it turned out that legally, the ITU, ISO, or IEC patent policy did not permit a royalty-free (RF) baseline with royalty-bearing (RB) options. Only Fair, Reasonable, and Non-Discriminatory (FRAND) terms were permitted for the whole standard. This has led to some patent litigation cases, which diminished between 2005 and 2006 when all the argued patents were running out. However, for future similar standardization projects, ideally, a Standards Developing Organization with a mixed RF and FRAND patent policy would be required.

DC-AC Prediction

A vital part of image compression is de-correlation. DCT is close to optimal for de-correlating the values within the 8×8 pixel blocks. In the standard, the DC value of the preceding block is used as the predictor for the current block. During the development of JPEG, a scheme for a more advanced inter-block de-correlation, using AC prediction, was considered. Based on the DC values of neighboring blocks, AC values in the center block can be predicted. However, JPEG has not integrated this scheme due to increased complexity. Instead, it was suggested as a decoder-only option.

Lossless

Most early JPEG research efforts went into the development of the higher compression (lossy) mode. However, lossless coding was essential to certain applications, as was a JPEG requirement.

Even though integer DCT would have provided the first choice, straightforward differential pulse code modulation (DPCM) was chosen by JPEG for the lossless mode. DPCM is applied in the pixel domain, where the value of a given pixel in a given color component is represented by the difference between the true value and a predicted value, and then compressed with a straightforward entropy coding technique (Huffman). Seven DPCM predictors are defined in the standard.

Figure 6. The JPEG logo. The ISO/IEC-ITU JPEG image compression standard is celebrating its 25th anniversary of approval. Throughout the years, other milestones of the standard have been celebrated at various events, such as at EPFL, Lausanne, Switzerland in 2013 and at a town-hall of Cesson-Sévigné, Rennes, France in 2014 and at the cultural center of Saint-Malo, France and at the University of Leipzig, Germany in 2016.

With real-life images, the compression can vary substantially (25–30 percent) with the choice of predictor. Typical compression factors between 2 and 3 can be obtained, depending on the complexity of the image and, notably, the pixel noise in the image. JPEG LS, based on the LOCO-I algorithm, was standardized with Huffman coding in 1999 and with extensions such as arithmetic coding in 2003. JPEG LS can typically give compression factors better than four.

At its creation, the goal of JPEG was a common compression scheme able to handle bi-level, halftone, and natural images. However, following the subjective testing of images, it was agreed that bi-level and halftone images would need a specific compression scheme. That gave birth to the Joint Bi-level Image Group (JBIG) in 1988, which resulted in ITU-T Recommendations ITU-T.82 (1993) and T.88 (2000). Later, the JPEG2000 project also fulfilled this "original" (but given up) JPEG requirement.

The ISO/IEC-ITU JPEG image compression standard is celebrating the 25th anniversary of the approval of the JPEG standard (see Figure 6). JPEG was the first international standard adopted for compression of natural tone digital images.[12] It is remarkable, even to those involved in its creation, that the compression technique has shown such resilience, providing the foundation for future extensions of JPEG, including JPEG2000, JPEG XT, and High Efficiency Video Coding (HEVC)-intra. **MM**

References

1. *ITU-T T.81 Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines*, Int'l Telecommunication Union, 1992.
2. *ITU-R Recommendation 601: Encoding Parameters of Digital Television for Studios*, Int'l Telecommunication Union, Feb. 1982.
3. G. Hudson et al., "PICA-Photovideotex Image Compression Algorithms," *ESPRIT '86: Results and Achievements*, Elsevier, 1987, pp. 413–421.
4. A. Léger, H. Poulsen, and M. Guglielmo, "Still Picture Compression Algorithm for Photovideotex on ISDN," *Proc. 5th Ann. Esprit Conf.* (ETW), 1988, pp. 1101–1108.
5. A. Léger, T. Omachi, and G. Wallace, "The JPEG Still Picture Compression Algorithm and Its Applications," *SPIE J. Optical Engineering*, vol. 30, no. 7, 1991, pp. 947–954.
6. W.B. Pennebaker, J. Vaaben, and J. Mitchell, "Progressive Coding in the Emerging JPEG Compression Standard," *Proc. Picture Coding Symp.* (PCS), Mar. 1990, pp. 26–28.
7. A. Léger, "Implementation of Fast Discrete Cosine Transform for Full Color Videotex Services and Terminals," *Proc. IEEE Global Telecommunications Conf.* (Globecom), 1984, pp. 333–337.
8. H. Poulsen, R. Vivian, and G. Wallace, "Subjective Testing Results for Still Picture Compression Algorithms for International Standardization," *Proc. IEEE Global Telecommunications Conf.* (Globecom), 1988, pp. 1022–1027.
9. G. Hudson, H. Yasuda, and I. Sebestyen, "The International Standardization of a Still Picture Compression Technique," *Proc. IEEE Global Telecommunications Conf.* (Globecom), 1988, pp. 1016–1021.
10. A. Léger, J. Mitchell, and Y. Yamazaki, "Still Picture Compression Algorithms Evaluated for

International Standardization," *Proc. IEEE Global Telecommunications Conf.* (Globecom), 1988, pp. 1028–1032.
11. H. Lohscheller, "A Subjectively Adapted Image Communication System," *IEEE Trans. Communications*, Dec. 1984, pp. 1316–1322.
12. W.B. Pennebaker and J.L. Mitchell, *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold Publishers, 1992.

Graham Hudson was a multimedia development manager at British Telecom Research Laboratories when working on the JPEG standard. He was the first chairman of ISO JPEG and chairman of the European PICA project. Contact him at graham.p.hudson@gmail.com.

Alain Léger was head of the research lab in knowledge processing at France Telecom Labs (Direction of Research), but when working on the JPEG standard, he was head of the videotex image coding group. He was chairman of the ISO JPEG-ADCT group. Contact him at a-t.leger@wanadoo.fr.

Birger Niss was a systems developer of the Copenhagen Telephone Company when working on the JPEG standard. He was a member of the image coding/interpretation ISO JPEG-ADCT group in 1985. Contact him at birger.niss@mail.tele.dk.

István Sebestyén is the Secretary General of Ecma International. When working on the JPEG standard, he was the chief engineer at Siemens and the CCITT special rapporteur for New Image Communication. Contact him at istvan@sebestyen.de.

This article originally appeared in
IEEE MultiMedia, *vol. 24, no. 2, 2017.*

THE SILVER BULLET
SECURITY PODCAST WITH GARY McGRAW
IEEE SECURITY&PRIVACY
SYNOPSYS®

This series of in-depth interviews with prominent security experts features Gary McGraw as anchor. *IEEE Security & Privacy* magazine publishes excerpts of the 20-minute conversations in article format each issue.

www.computer.org/silverbullet
***Also available at iTunes**

Sally Weber: Making Art from Light

Sally Weber
Resonance Studio

Bruce Campbell
Rhode Island School of Design

Francesca Samsel
University of Texas—Austin

Editors:
Bruce Campbell
bcampbel01@risd.edu

Francesca Samsel
figs@cat.utexas.edu

We caught up with Sally Weber (www.sallyweber.com) after having been transfixed by experiencing her latest work, inFLUX, in her exhibition, ELEMENTAL (a current installation) at the Butler Institute of American Art, in Youngstown, OH. As an artist who has worked with light as her medium for a distinguished career, we believe Sally has much of value to share with our readership.

Weber is a holographic artist who completed her graduate work at the Center for Advanced Visual Studies at MIT under Otto Piene. While at MIT, she was able to explore holography and produce several solar installations through the facilities of the Regional Laser Center in the George R. Harrison Spectroscopy Laboratory.

Like graphic artists and data visualization specialists who use models of light and color to present content for screen and paper, Sally works to communicate with light but through media such as holography and laser traces. She works to find a resonance in the materials and representations in order to provide unique experiences of light to elicit insight regarding phenomena not easily experienced at human spatial and temporal scale. Her iterative process includes exploring tangible materials to find an art piece composition that can transfix an audience—an experiential state of being that results in lingering long enough to appreciate the art while wondering about the underlying communication.

Upon experiencing her art, one can think through the aspects of her pieces that make the experience of them so compelling and what technologies might possibly need to provide as affordances to make such experiences transfixing using virtual reality technologies.

Through her environmental art Sally took on the challenge of how to bring art into a larger environment, by working from the outside in. She studied the nature of a place and then made art that was site specific—back when public art was just starting to come to light. Seattle was one of the first locations that had a public art program and by the time she moved out to California, she was right in the middle of it. In doing larger architectural scale work—using sunlight to illuminate holographic art installations—she didn't think of a hologram as a little thing. She thought of landscape and the idea of bringing light and color into a building so that it became a time piece—a way to capture an experience of something that was always in motion. Examples of earlier work include *Lightscape* (Figure 1) and *FocalPoint* (Figure 2)

 Published by the IEEE Computer Society 2469-7087/18/$33.00 © 2018 IEEE

Figure 1. *LightScape,* is a solar installation using holographic optical elements and acrylic. Sited on Kresge Lawn, Massachusetts Institute of Technology, Cambridge, Massachusetts. Designed to reflect the sun's arc across the site and Kresge Auditorium's curved roof. The holographic elements float above the ground and alter in color with the sun's angle and the viewer's distance from the installation. Photo credit: Sally Weber. (Used with permission.)

Figure 2. *FocalPoint.* is a solar water fountain, 12'x 4'x 8', with holographic optical elements, glass pipes, steel, and a water system. Designed to focus sunlight into 3 lines of light that scan across the floor and walls in response to the sun's motion. Installed here at the Boston Museum of Science overlooking the Charles River, Boston, Massachusetts. Photo credit: Sally Weber. (Used with permission.)

Weber's work is complex and wide ranging in media and content. Focusing on her current exhibition *inFLUX* below, Weber speaks of the work, the process and underlying principles in her own words:

"The themes within *inFLUX* have been percolating for over many years. I had worked with lasers and optics for quite a number of years and knew it wasn't a holography piece but one that could get to the jitter—the inherent jitter that is in everything. Jitter, like that in Brownian motion, is the constant movement of molecules. If you have a glass of water and you pour some milk into it, over time the whole glass of water will become milky. All the molecules bouncing off of each other are actually doing the mixing. For me it doesn't matter what scale—it's a metaphor in that everything is jittering. You can see images from the Large Hadron particle collider at CERN and that's what they are trying to capture—the motion that is a result of collisions which defines what particles exist and what's going on with them as a result of the collisions.

"Light as photons, the particles of light, are always in motion. People generally tend to think of light as just all pervasive, but it is dimensional. Light is always moving and we go through this matrix of movement—affecting not just us but everything. *InFlux* became a metaphor for that kind of movement—not as a rhythmic kind of pendulum motion, like to and fro, that we think of as its resonance dependent on the length of the pendulum, but the movement of objects that interact with each other—a physical representation of what is happening all the time, independent of scale. We 'jitter' or interact with each other in some of the same ways when we meet up with someone, have a conversation, walk by them, etc. The way we move, it's a dance, a synchronicity that takes place all the time, unconsciously, and is something that has been roaming around in my head for a long time.

"I knew I wanted the light to be the functional aspect of *inFLUX*, the active ingredient doing the drawing and leaving a trace of its path, so that people could find a place where they were just caught—and in art when you get caught there can be a moment of silence. You are caught where you aren't thinking and you aren't just emoting, but are transfixed as if through to the solar plexus. As an artist, occasionally, I hope to catch someone before words. In that place there is experience. I aim for that ongoing sense of wonder when one has to stop long enough to see and feel. Otherwise it goes right past you or you go right past it.

"There is a concept in Buddhism that is not a process of cause and effect but when everything arises at once. If one thinks of nature as cyclic, instead of linear, with everything arising at once, you can't pull it apart. So if instead of looking at the separate parts, which science has done to be able to consider parts separately in order to delve into it, you are looking at the whole matrix of the web and the structure in between it as if you are looking at the negative space. I think I have been fascinated by what keeps things together and how one part impacts the other. So to make things that naturally have limits, you try to expand those limits so that people's experiences take them to someplace else.

"You can take the experience of a total eclipse and try to capture it through pictures but that isn't going to do it. It's because the whole world that you know changes in a matter of moments when you finally get to totality.

"A few years ago an astronomer said to me, '*for a photon it is always present.'* I loved that. You take that sentence in and you think you get it, and you don't quite get it at the same time. That paradox is exactly that place requiring lingering—the tension between the known and the unknown is right there. For me, light is everything from looking at a star and being right there while at the same time you are right here, as well as thinking of light not just as a surface, or a totality of reflected surfaces, but as the stuff out here, *in the air between us.* From the start holography offers a unique medium: to be able to bring light to a conceptual mode where you think of it as an object but it is not, it is really a field of light. It's taking the physicality away but giving light a sense of boundary and that sense of boundary makes us ask what else is just a boundary that we consider solid? And thinking about the permeability between these media and how delicate that is or how robust.

Figure 3. Laser pendulums draw in the sand leaving a trace of their path as color patterns building up over time as *InFLUX*, a two-part laser pendulum installation, in *ELEMENTAL*, solo exhibition at the Butler Institute of American Art, Youngstown, Ohio. Photo credit: Sally Weber. (Used with permission.)

"If you can do something that just shakes people a little bit it's as if they are looking at a new vista. They suddenly ask "oh my gosh, what is that?" In the case of leaving traces of light in the sand and people not knowing why, it allows that expansion.

"Little kids are great—they will kneel down spontaneously to look closely. Adults get caught up in the motion and look up to see what is causing it while the kids don't bother—because they are watching the motion and color and hearing the sounds whirling with the twisting and twisting of the actions that you can't take all in at once. And yet it's leaving a trace by drawing so you can see it over time—building up layers of these colored patterns that gradually fade and that are illuminated by the lasers and then glow.

"Physically the exhibit consists of white sand down on the floor and the pendulums above it (see Figure 3). I watch where they go and adjust things accordingly and I put what I call *magic dust*, that is basically different pigments, down on the sand so you can see glow in different colors. Depending on when the lights are low or are off, you really see how over time they build up. They remind me of nebula because of the same kind of looking into the depths of something and trying to ascertain from photographs what is in front and what is in back—the brightness is obviously on top here but there is this complexity underneath. That complexity is all that the light drew over time. It is gradually fading or being reinforced continually."

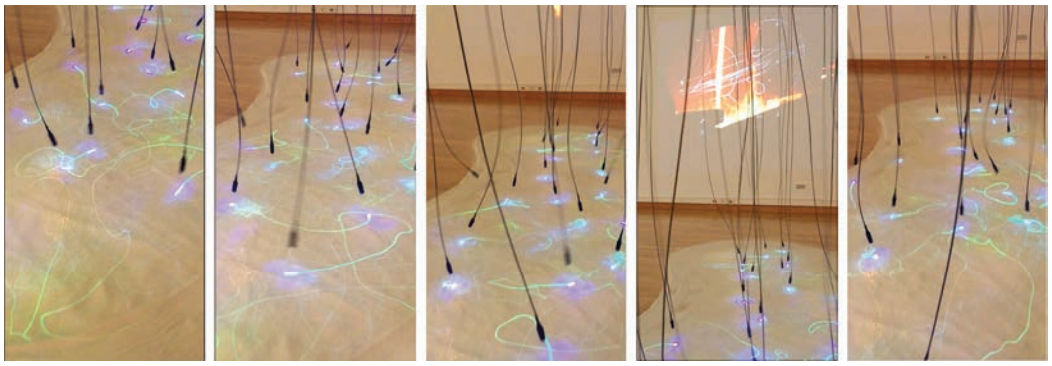


Figure 4. A series of stills from *inFLUX* at the Butler Institute of American Art. Each laser draws in the sand by illuminating pigments, which glow in response to the light. Over time, patterns emerge and fade to be redrawn as the pendulums interact, collide, twist and pass by each other. Photo credit: Sally Weber. (Used with permission.)

Selected frames of *inFLUX* in action are shown in Figure 4. A video of *inFLUX* can be found at (URL). For further information about Sally Weber and her work, see www.sallyweber.com.

ABOUT THE AUTHORS

Sally Weber is an independent artist at Resonance Studio. She received her Master's degree in Art & Visual Studies from Massachusetts Institute of Technology. Her research interests include applying holography and laser technologies in support of art and visualization. Contact her at info@sallyweber.com.

Bruce Campbell is faculty within the Web Design + Interaction department at the Rhode Island School of Design. He received his PhD in systems engineering from the University of Washington after a professional period pursuing information science application to complex information spaces. His research interests include data visualization and user-centric interaction techniques. Contact him at bcampbel01@risd.edu.

Francesca Samsel is a research associate at the University of Texas–Austin. She received her MFA in sculpture from the University of Washington. Her research interests include the intersections between art science and technology. Contact her at figs@cat.utexas.edu.

Contact department editor Bruce Campbell at bcampbel01@risd.edu or department editor Francesca Samsel at figs@cat.utexas.edu.

This article originally appeared in
IEEE Computer Graphics and Applications,
vol. 38, no. 3, 2018.

Editor: **Cesare Pautasso**
University of Lugano
c.pautasso@ieee.org

Editor: **Olaf Zimmermann**
University of Applied Sciences
of Eastern Switzerland, Rapperswil
ozimmerm@hsr.ch

A Taxonomy of IoT Client Architectures

Antero Taivalsaari and Tommi Mikkonen

From the Editors

The Internet of Things (IoT) makes the world programmable. IoT software embeds numerous resources, from sensors and actuators on the edge, all the way to automation platforms in the cloud. Antero Taivalsaari and Tommi Mikkonen share their project experiences at Nokia Health and Mozilla Connected Devices, distilled into different stacks of increasing complexity for deploying such software across many different kinds of Things. —*Cesare Pautasso and Olaf Zimmermann*

AT THE TECHNICAL level, the Internet of Things (IoT) is all about turning physical objects and everyday things into digital data products and services—bringing new value and meaning by making previously lifeless things more intelligent. Effectively, this means adding computing capabilities and cloud connectivity to hitherto unconnected devices, as well as adding back-end services and web or mobile apps for viewing and analyzing data and controlling those devices.

IoT systems are end-to-end systems consisting of four basic architectural elements that tend to be pretty much identical in all IoT solutions.[1] *Devices* are the physical hardware elements that collect sensor data and might perform actuation. *Gateways* collect, preprocess, and transfer sensor data from devices and might deliver actuation requests from the cloud to devices. The *cloud platform*—usually offered as a software-as-a-service solution—has a number of important roles, including data acquisition, data analytics, and device management and actuation. *Applications* range from simple web-based data visualization dashboards to highly domain-specific mobile apps.

A wide spectrum of software architecture options exists for IoT devices, ranging from very simple, limited sensing devices to devices featuring fully fledged OSs and developer APIs. In this article, we define a simple taxonomy of these options based on a number of industrial and academic IoT development projects carried out in the past four years. (For two examples, see health.nokia.com/es/en/steel-hr and wiki.mozilla.org/Connected_Devices/Projects.)

Software Architecture Options

IoT systems involve various design drivers and tradeoffs. Important factors include cost, update capabilities, dynamic programmability, security, energy efficiency, and communication latency. These factors largely determine the architecture options we describe next.[2]

On a high level, the software architecture choices for IoT client devices fall into the following seven categories, ranging from simple to more complex:

- *no-OS architectures*,
- *RTOS (real-time OS) architectures*,
- *language-runtime architectures*,
- *full-OS architectures*,
- *app-OS architectures*,
- *server-OS architectures*, and
- *container-OS architectures*.

2469-7087/18/$33.00 © 2018 IEEE Published by the IEEE Computer Society

No-OS Architectures

The vast majority of today's IoT devices are really simple. Smart light bulbs, thermostats, remotely controlled electricity plugs, air quality sensors, and ID tags or badges don't require complex software stacks.

Such simple IoT devices don't need an OS or application platform. All the software is written specifically for the device, and software development typically is in-house. So, third-party-developer support is unnecessary. Support for firmware updates might be limited or nonexistent.

Given the fixed nature of software in such low-end devices, the amount of RAM and flash memory can be minimized. In many cases, only a few kilobytes or tens of kilobytes of RAM will suffice.

For battery-operated low-end devices, network communication optimization plays a major role. Communication protocols such as MQTT (MQ Telemetry Transport), LWM2M (Lightweight Machine-to-Machine), and CoAP (Constrained Application Protocol) are important, whereas more-capable devices tend to use HTTP-based communication and more verbose data formats such as JSON (JavaScript Object Notation) or XML.

RTOS Architectures

For slightly more capable devices supporting a richer set of sensors, an RTOS might be beneficial. Popular open source and commercial RTOSs provide convenient developer toolkits and a basic set of APIs supporting second-party software development. They also support important product features such as secure firmware updates.

Software development for RTOS-based IoT devices is usually in-house because such devices typically don't provide public third-party-developer APIs or the ability to reprogram the device dynamically (apart from performing a full firmware update). Typical development languages for RTOS-based devices are C or C++, although even assembly code might be used in some areas.

The memory requirements of RTOS-based architectures are comparable to no-OS architectures, often necessitating as little as a few tens of kilobytes of RAM and a few hundred kilobytes of flash memory. Devices in this category are often battery-operated, thus placing many requirements on optimizing network connectivity and energy consumption more broadly.

Language-Runtime Architectures

Some IoT development boards support a specific built-in language runtime or virtual machine (VM). For instance, the popular Espruino (www.espruino.com) and Tessel 2 (tessel.io) boards support JavaScript applications, while Pycom's WiPy boards (pycom.io/development-boards) enable Python development.

Compared to no-OS or RTOS solutions, language-runtime-based IoT devices are significantly more capable. They can support third-party application development and dynamic changes—updating the device software (or parts thereof) dynamically without having to reflash the entire firmware.

At the conceptual and technical levels, language-runtime-based IoT devices are very similar to early mobile-app development platforms such as the Java 2 Platform, Micro Edition (J2ME). In J2ME, a dynamic language runtime served as the portable execution layer that enabled third-party application development and the creation of developer-friendly application interfaces. Such capabilities leverage the interactive nature of the dynamic languages, allowing flexible interpretation and execution of code on the fly, without compromising the security of the underlying execution environment and device. Basically, applications run in a sandbox that provides only limited access to the underlying platform features.

At the implementation level, language-runtime-based IoT devices typically have an RTOS underneath. In that sense, these devices can be seen as the next evolutionary step up from devices built on the RTOS architecture.

The technical capabilities and memory requirements of devices based on a language-runtime architecture vary considerably on the basis of the supported languages. The VMs' size and complexity also vary considerably. Minimalistic programming languages such as Forth might require only a few tens of kilobytes of dynamic memory, whereas Python or JavaScript VMs require at least several hundreds of kilobytes or preferably multiple megabytes of RAM. Correspondingly, the minimum amount of flash or ROM memory can also range from a few tens of kilobytes to several megabytes. However, storage memory is now so inexpensive that its cost only marginally affects a device's total price.

Full-OS Architectures

The next level up from a language-runtime architecture is IoT devices that are powerful enough to run a full (typically Linux-based) OS. The Raspberry Pi 3 is a great example of such a device.

The presence of a full OS brings many benefits, such as built-in support for secure file transfers, user

accounts, device management, security updates, mature development toolchains, and numerous other features. The generic nature of devices supporting a full-OS architecture also makes it possible to effortlessly run various types of third-party applications and services, including the aforementioned language runtimes.

Compared to no-OS or RTOS architectures, full-OS stacks have significantly higher memory and CPU requirements. For instance, the desire to run a Linux-based OS in a device bumps the RAM requirements from a few tens or hundreds of kilobytes (for an RTOS-based solution) to half a megabyte at a minimum. The significantly higher energy consumption requirements make it difficult to employ such devices in use cases that require battery operation—except in tablet- or laptop-sized solutions with a battery capacity of at least a few thousand milliampere hours.

App-OS Architectures

At the current high end of the IoT device spectrum are wearable-device platforms such as Android Wear (www.android.com/wear) and Apple watchOS (www.apple.com/watchos). These platforms are in many ways comparable to mobile-phone-app platforms from three to five years ago. They provide rich platform capabilities and third-party-developer APIs; however, they also bump up the minimum hardware requirements considerably. For instance, Android Wear and watchOS require a minimum of half a gigabyte (512 Mbytes) of RAM—over 10,000 times more than the few tens of kilobytes of RAM required for simple IoT sensor devices.

The processing-power requirements of app-OS devices are also dramatically higher than in the simplest microcontroller-based IoT devices. Typically, an ARM Cortex-A class processor is mandated. (For instance, Android Wear currently requires, at a minimum, an ARM A7 processor running at 1.2 GHz.) This limits the maximum battery duration to a few days, or only a few hours in highly intensive use.

Server-OS Architectures

Much to nearly everybody's surprise, JavaScript surpassed the other programming languages in popularity in 2016.[3] Whereas JavaScript was originally designed in the mid-1990s as a simple scripting language for web browsers, in recent years its use has rapidly spread into various other areas. Its current success can be attributed especially to the Node.js ecosystem (nodejs.org), which has popularized the use of JavaScript in server-side development too. Thus, JavaScript has become the lingua franca for web development from client to cloud.

The popularity of Node.js has created interest in IoT devices that can host a webserver. For instance, the Tessel 2 board can run the Node.js stack and even serve as a standalone webserver. Similarly, Raspberry Pi devices are commonly used for running the Node.js stack and other webservers.

By default, Node.js assumes the availability of at least 1.5 Gbytes of RAM. However, it can be configured to operate with considerably less memory, starting from a few tens of megabytes. Besides Node.js, there are several other webserver offerings that are more tailored to embedded environments.

Container-OS Architectures

Container-based software architectures have recently become popular, especially in cloud back-end development.[4] A container is a standalone, portable, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, and settings. Popular implementations include Docker and CoreOS rkt.

Containers isolate applications from one another and the underlying OS infrastructure, while providing an added layer of protection for the application. This guarantees that the software will always run the same way regardless of its physical execution environment.

At the technical level, containers are effectively a lighter-weight OS virtualization mechanism. Unlike OS VMs such as VirtualBox or VMware Workstation, containers don't virtualize a complete guest OS but share the underlying OS with other containers.

Given the independence of the physical execution environment that containers can provide, they're also an attractive choice for IoT development, especially in light of IoT devices' current technical diversity. Thus, although container technologies add considerable overhead compared to traditional binary software, they're already being used with IoT devices. For instance, Docker can already be used on Raspberry Pi devices.[5]

From a purely technical viewpoint, container-based architectures are definitely a viable option for IoT devices if adequate memory and other resources are available.[4] At a minimum, the host environment typically must have several gigabytes of RAM available, thus making this approach unsuitable for the vast majority of today's IoT devices. Although container-based IoT devices might seem excessive today, we see them as

Table 1. Software architecture options for IoT devices.*

Feature	Architecture option					
	No OS or RTOS	Language runtime	Full OS	App OS	Server OS	Container OS
Typical devices	Simple sensor devices, heartbeat sensors, lightbulbs, and so on	Feature watches, more advanced sensing devices	"Maker" devices, generic sensing solutions	High-end smartwatches	Solutions benefiting from a portable webserver and edge-computing capabilities	Solutions benefiting from fully isomorphic apps—that is, code that can be migrated between the cloud and the edge
Minimum required RAM	Tens of kilobytes	Hundreds of kilobytes	A few megabytes	Hundreds of megabytes	Tens of megabytes	Gigabytes
Typical communication protocols	Constrained (MQTT, LWM2M, CoAP)	Constrained (MQTT, LWM2M, CoAP)	Standard Internet protocols (HTTP, HTTPS)	Standard Internet protocols (HTTP, HTTPS)	Standard Internet protocols (HTTP, HTTPS)	Standard Internet protocols (HTTP, HTTPS)
Typical development language	C or assembly	Java, JavaScript, Python	C or C++	Java, ObjectiveC, Swift	JavaScript	Various
Libraries	None or system-specific	Language-specific generic libraries	OS libraries, generic UI libraries	Platform libraries	Node.js npm modules	Various
Dynamic software updates	Firmware updates only	Yes	Yes	Yes	Yes	Yes
Third-party apps supported	No	Yes	Yes	Yes	Yes	Yes
Isomorphic apps possible	No	Yes	Only if the hardware architectures are binary compatible	Yes	Yes	Yes

** RTOS = real-time operating system, VM = virtual machine, MQTT = MQ Telemetry Transport, LWM2M = Lightweight Machine-to-Machine, and CoAP = Constrained Application Protocol.*

an important step toward fully isomorphic IoT system architectures, which we discuss in the next section.

Observations and Trends

The selection of a software architecture for IoT devices depends on the expected use, the power budget, and the need to support dynamic programming or third-party development. Table 1 summarizes the architecture options. In general, the more capable the underlying execution environment is, the more feasible it is to run various types of software architectures, platforms, and applications on it.

In particular, we make the following six observations.

First, energy consumption requirements heavily influence the software architecture choice. In practice, one of the most significant differentiating features driving or even dictating the selection of the software architecture in most IoT devices is the battery. A battery-operated IoT device typically has strict minimum operating-time requirements. Furthermore, a device's form factor plays a significant role in determining the right trade-offs, thus also impacting the type of software architecture the device can support.

Second, the availability of inexpensive off-the-shelf hardware is

driving the industry toward "overly capable" IoT devices. That is, the recent emergence of inexpensive IoT chips, development boards, and maker kits is leading to IoT devices and solutions that have more processing power and memory than are actually needed. Given this, it might often be simpler and much more affordable to buy stock hardware instead of building custom hardware solutions. Furthermore, the extra capacity can be beneficial—for example, for improved security features.

Third, software development for IoT devices is very similar to classic embedded-systems development and is thus bringing back the need for embedded-software-development skills and education.[6] This is in contrast with recent software industry survey reports that emphasize the importance of higher-level programming skills.[7]

Fourth, in the next 5 to 10 years, the availability of software containers and virtualization technologies in IoT devices will lead the industry to isomorphic IoT system architectures. In these architectures, the devices, gateways, and cloud will be able to run exactly the same software components and services. This will allow flexible migration of code between any element in the overall system. In such an architecture, there don't have to be any technical differences between software that runs on the back end or on the network edge. Rather, when necessary, software can freely roam between the cloud and the edge in a seamless, liquid fashion.

Fifth, along the way toward isomorphic systems, edge computing will play an increasingly important role. Given IoT devices' rapidly increasing computing and storage capacities, it's clear that in the future, computation and intelligence will be increasingly balanced between the cloud and the edge (IoT devices and gateways). This could be very beneficial because the ability to preprocess data in IoT devices (and gateways) allows for lower latencies and can significantly reduce unnecessary data traffic between the devices and the cloud. Together with the emergence of mesh networking and low-power wide-area networking (LPWAN) technologies, edge computing can be expected to significantly alter the topologies and overall software architecture of IoT systems.

Finally, interoperability is still a major issue. Today, most IoT systems expect that devices will work only with their "own" cloud back end. Similarly, the most common way to use a device is through a specific application that's associated with only one particular vendor's devices. Even though significant convergence has occurred in the past few years, we're still several years away from universal Programmable World standards as envisioned by Bill Wasik[8] and discussed in our previous *IEEE Software* article.[1]

According to a popular saying—often attributed to Mark Twain—history does not repeat itself, but it rhymes. At the moment, there's still much diversity in the IoT device space. In many ways, the IoT device market today resembles the early evolution of the PC market before the dominant PC platforms were established in the early 1980s. Interesting parallels also exist between today's IoT devices and the evolution of mobile phones in the late 1990s and early 2000s. Although the vast majority of IoT devices today have very simple software stacks, we foresee stack complexity increasing rapidly because of hardware evolution and the

ABOUT THE AUTHORS

ANTERO TAIVALSAARI is a Bell Labs Fellow at Nokia Technologies. Contact him at antero.taivalsaari@nokia.com.

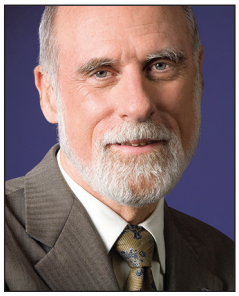
TOMMI MIKKONEN is a professor of software engineering at the University of Helsinki. Contact him at tommi.mikkonen@helsinki.fi.

general desire to support edge computing, software containers, and isomorphic IoT system architectures.

References

1. A. Taivalsaari and T. Mikkonen, "A Roadmap to the Programmable World: Software Challenges in the IoT Era," *IEEE Software*, vol. 34, no. 1, 2017, pp. 72–80.
2. D. Spinellis, "Software-Engineering the Internet of Things," *IEEE Software*, vol. 34, no. 1, 2017, pp. 4–6.
3. D. Cassel, "JavaScript Popularity Surpasses Java, PHP in the Stack Overflow Developer Survey," *The New Stack*, 18 Mar. 2016; thenewstack.io/javascript-popularity-surpasses-java-php-stack-overflow-developer-survey.
4. A. Celesti et al., "Exploring Container Virtualization in IoT Clouds," *Proc. 2016 IEEE Int'l Conf. Smart Computing* (SMARTCOMP 16), 2016.
5. M. Richardson, "Docker Comes to Raspberry Pi," blog, 30 Aug. 2016; www.raspberrypi.org/blog/docker-comes-to-raspberry-pi.
6. C. Weir and J. Noble, *Small Memory Software: Patterns for Systems with Limited Memory*, Addison-Wesley, 2000.
7. *Developer Economics State of the Developer Nation Q1 2016*, SlashData, 2016; www.developereconomics.com/reports/developer-economics-state-of-developer-nation-q1-2016.
8. B. Wasik, "In the Programmable World, All Our Objects Will Act as One," *Wired*, 14 May 2013; www.wired.com/2013/05/internet-of-things-2.

This article originally appeared in IEEE Software, *vol. 35, no. 3, 2018.*



A Revised View of the IoT Ecosystem

Vinton G. Cerf • *Google*

This article originally appeared in IEEE Internet Computing, *vol. 21, no. 5, 2017.*

For a long time, I had the idea that configuration of a suite (ensemble) of Internet of Things (IoT) devices would be an infrequent process — for example, when you bought a new device, sold, transferred, or superannuated an older one. However, my thinking has evolved. Sticking with the residential paradigm for a moment, although the ideas seem equally applicable to industrial settings, it's becoming clearer that many devices will come and go with the residents, guests, workmen, emergency services personnel, and others who might have reason to enter the premises and have need to "control" at least a part of it while present.

This leads me to believe that an IoT ensemble must actually be in a kind of continuous configuration mode, anticipating the arrival and departure of all manner of Internet-enabled devices. Among the implications is the notion that the local IoT management system needs to expect that new devices will need to be configured into the system and others to depart — it needs to sense their arrivals and departures and to react accordingly.

Not every device that arrives must be configured into the system, nor must every device that leaves be deconfigured. Indeed, some devices must be recognized when in remote locations, to be authoritative with regard to access to data and ability to exert controls. Others should be ignored even when on the premises. This implies that there must be a highly active process for discovering and qualifying devices to become part of the local IoT ecosystem and to be recognized as authoritative even when not local.

By extension, these devices must be able to present bona fides to the residential IoT control system when called upon to do so. The process must be painless for users, but assure household authorities that only devices (and people) that should be granted access are properly identified. This strikes me as a nontrivial design challenge; the ecosystem will need some serious thinking about standards to achieve interoperability across a multitude of potential "players" that might be encountered.

The Bluetooth technology-pairing mechanism offers an example of device discovery and a means of confirming that a selected device should become associated with another. For example, cars equipped with Bluetooth technology can detect the presence of another Bluetooth device if the latter is put into a beaconing mode. The car typically serves as the master and discovers a beaconing slave. The master sends the slave a locally generated random number, typically displayed on the slave device. Users are asked to verify that both the master and slave are displaying the same random number before the master adopts the slave. Protocols like this are already in use to allow controllers to incorporate new IoT devices into an ensemble. In a residential setting, we can easily imagine a home controller that detects and configures new devices into its universe, and that can be told to forget an adopted slave when it should be deconfigured (upon the departure of a visitor, for example) or to remember the device and to have a means to recognize it again even when it's remote and communicating — for example, via the Internet.

Here's a scary thought: what if a device is adopted that's corrupted, and it has a backdoor allowing remote access to a residential network of devices?

Vinton G. Cerf is vice president and chief Internet evangelist at Google, and past president of ACM. He's widely known as one of the "fathers of the Internet." He's a Fellow of IEEE and ACM. Contact him at vgcerf@gmail.com.

SUBMIT TODAY

IEEE TRANSACTIONS ON BIG DATA

SCOPE

The *IEEE Transactions on Big Data* (*TBD*) publishes peer reviewed articles with big data as the main focus. The articles provide cross disciplinary innovative research ideas and applications results for big data including novel theory, algorithms and applications. Research areas for big data include, but are not restricted to, big data analytics, big data visualization, big data curation and management, big data semantics, big data infrastructure, big data standards, big data performance analyses, intelligence from big data, scientific discovery from big data security, privacy, and legal issues specific to big data. Applications of big data in the fields of endeavor where massive data is generated are of particular interest.

SUBSCRIBE AND SUBMIT

For more information on paper submission, featured articles, calls for papers, and subscription links visit:

www.computer.org/tbd

IEEE computer society | IEEE COMMUNICATIONS SOCIETY | IEEE Computational Intelligence Society | IEEE Signal Processing Society | Systems, Man, & Cybernetics Society

IEEE VTS | CSS | IEEE photonics SOCIETY | EMB | IEEE PES Power & Energy Society | IEEE Biometrics Council | IEEE

IEEE COMPUTER SOCIETY: Be at the Center of It All

IEEE Computer Society membership puts you at the heart of the technology profession—and helps you grow with it.

Here are 10 reasons why you need to belong.

IEEE Computer Society—keeping you ahead of the game. Get involved today.

www.computer.org/membership

IEEE computer society