COMMUNICATIONS OF THE ACM

CACM.ACM.ORG

10/2026 VOL.69 NO.10



AI and the Future of Accessible Work

Governing Social Media as a Public Utility

Toward Agentic Runtime Healing

10 Systemic Changes to Broaden Participation in Computing

Association for Computing Machinery

SIGGRAPH ASIA 2026
KUALA LUMPUR



SIGGRAPH ASIA 2026

WEAVING THE FUTURE

CONFERENCE
1 – 4 December 2026

EXHIBITION
2 – 4 December 2026

VENUE
Kuala Lumpur Convention Centre (KLCC), Malaysia

Register Now!

ASIA.SIGGRAPH.ORG/2026

SPONSORED BY

ORGANIZED BY

koelnmesse

COMMUNICATIONS



Want to lead the news instead of following it?

Communications is searching for a chair or co-chairs to direct its News Board. The chair(s) will work with a diverse, worldwide board to identify coverage topics, assist professional writers in identifying appropriate sources, and review articles to ensure quality and accuracy.

If you are interested in this position, which does not require a Ph.D., please send a short statement describing your background and explaining why and how you would find new ideas for news stories to **eic@cacm.acm.org**.

Association for Computing Machinery

COMMUNICATIONS OF THE ACM

News



13 **The End of the Coder?**
As AI agents transition from simple coding assistants to end-to-end engineers, the profession is shifting from writing syntax to auditing complex systems.
By Logan Kugler

16 **Beyond LLMs: A Post-Transformer World Emerges**
A focus of the next decade might be on teaching AI how to master reality and understand how things work in the physical world.
By Samuel Greengard

109 Careers

Last Byte

112 Upstart Puzzles
Cheating by the Rules
Calculating candidate options.
By Dennis Shasha

Association for Computing Machinery
Advancing Computing as a Science & Profession

Opinion

5 From the President
Computing Is Changing; How Should ACM Change with It?
By Elisa Bertino

7 Cerf's Up
Why We Need Occupancy Intelligence during Emergencies
By Vinton G. Cerf and Harsh Verma

9 Career Paths in Computing
From Absence to Advocacy: Shaping the Future for Women in Tech
By Oluwasefunmi 'Tale Arogundade

10 BLOG@CACM
Converting Text to SQL, and Computing to Fix Red Tape
Michael Stonebraker and Peter Baile Chen address issues with data warehouses, while Vinay K. Chaudhri and Patrick A. McLaughlin consider AI's role in regulations.

20 Opinion
Who Owns the Wai (Water)? AI, Power, and the Public Trust
From extractive infrastructure to AI in the public trust.
By Josiah Hester

Watch the author discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/who-owns-the-wai

25 Technology Strategy and Management
The ASML Monopoly: Platform and Ecosystem Leadership in EUV Lithography
Scratching the surface of the leading manufacturer of extreme ultraviolet lithography technology used for manufacturing integrated circuits.
By Michael A. Cusumano

Opinion

29 Kode Vicious
Pebbles on the Beach
Keeping hold of the big questions.
By George V. Neville-Neil

31 Opinion
Governing Social Media as a Public Utility
A case for sovereign digital infrastructure.
By Christoph Mueller-Bloch and Raffaele Fabio Ciriello

35 Opinion
Bounded Minds, Generative Machines
Envisioning conversational AI that works with human heuristics and reduces bias risk.
By Jiqun Liu

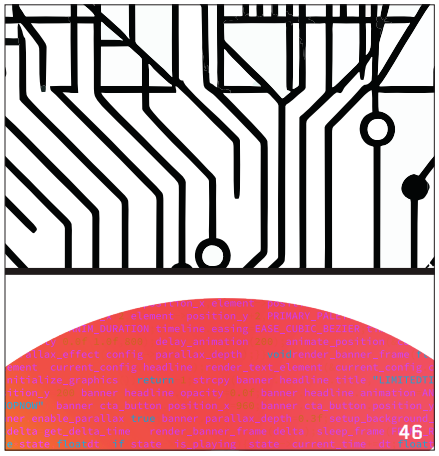
38 Opinion
The Multimodal Agent Interfaces Paradigm Shift
How the movement toward multimodal agent interfaces will fundamentally transform how users interact with digital systems.
By James A. Landay and Anoop K. Sinha

41 Opinion
ACM as a Professional Society: The ACM Volunteer Database
Expanding ACM's Volunteer Database could help members better document their varied volunteer service to the organization.
By Jeanna Matthews

43 Opinion
Everything Everywhere All-Pervasive Augmented Reality
Designing for and with people: A proposal for a pervasive augmented reality civics space.
By Holger Regenbrecht, Kushani Perera, and Nadia Pantidi

IMAGE FROM SHUTTERSTOCK.COM

Practice



46 **Instruction Set Migration at Warehouse Scale**
An analysis of how Google migrated a multibillion-line codebase from x86 to Arm.
By Eric Christopher, Kevin Crossan, Wolff Dobson, Chris Kennelly, Cannada Lewis, Kun Lin, Martin Maas, Parthasarathy Ranganathan, Emma Rapati, and Brian Yang

Research and Advances



58 **AI and the Future of Accessible Work**
Though AI offers exciting potential to improve workplace accessibility, it also poses significant risks and can create new barriers for disabled workers.
By Anne Marie Piper, Darren Gergle, Cynthia L. Bennett, Maitreya Shah, Chancey Fleet, Ali Abdolrahmani, Ariana Aboulafia, Stacy M. Branham, Sharon Koppman, Arielle Silverman, and Erik B. Sudderth

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/accessible-work

68 **Beyond the Hype: The Efficiency-Throughput Gap with GitHub Copilot**
A real-world field study examines how GitHub Copilot affects engineering productivity in an enterprise setting.
By Fei Liu and Todd McKinnon

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/beyond-the-hype

Research and Advances



78 **10 Systemic Changes to Broaden Participation in Computing**
Results from a 22-institution, multi-year project to broaden participation in computing show that implementing even a subset of the 10 systemic changes leads to increases in retention and attraction.
By Carla E. Brodley, Catherine Gill, Megan Giordano, and Caitlin J. Kidder

88 **Toward Agentic Runtime Healing**
LLMs can dynamically generate error-handling strategies in real time, tailored to specific runtime contexts such as error messages and program states.
By Zhensu Sun, Haotian Zhu, Bowen Xu, Xiaoning Du, Li Li, and David Lo

Research Highlights

98 Technical Perspective
NVMR for Batteryless Computing
By Brandon Lucia

99 **Non-Volatile Memory Renaming for Safe Batteryless Computing**
By Abhishek Bhattacharyya, Abhijith Somashekhar, and Joshua San Miguel



About the Cover: Looking into the future of AI for accessible work reveals just as many challenges as opportunities. Based on workshop discussions with disabled participants, this article highlights six areas—beyond well-known technical limitations such as biases and hallucinations—where AI can introduce unintended barriers to accessibility in the workplace, and what can be done to address them. Cover illustration by Peter Crowther Associates.

IMAGES L-TO-R BY: ANDRIJ BORYS ASSOCIATES, USING SHUTTERSTOCK.COM; PETER CROWTHER ASSOCIATES; NORTHEASTERN UNIVERSITY

COMMUNICATIONS OF THE ACM

Trusted insights for computing's leading professionals.

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

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

Executive Director and CEO
Alison Derbenwick Miller
Deputy Executive Director and COO
Patricia Ryan
Director, ACM Digital Library
Wayne Graves
Director, Office of SIG Services
Donna Cappo
Director, Office of Publications
Scott E. Delman

ACM COUNCIL

President
Elisa Bertino
Vice-President
Rashmi Mohan
Secretary/Treasurer
Tom Crick
Past President
Yannis Ioannidis
Chair, SGB Board
Neha Kumar
Co-Chairs, Publications Board
Wendy Hall and Divesh Srivastava
Members-at-Large
Odest (Chad) Jenkins, John Kim, Yunyao Li, Alejandro Saucedo, Lydia Tapia, Holly Yanco
SGB Council Representatives
Adrienne Decker, Jonathan Aldrich, and Vivek Sarkar

BOARD CHAIRS

Education Board
Briana Morrison and Thomas Cortina
Practitioners Board
Marlene Mhangami and Sophie Watson
Digital Library Board
Jack Davidson

TOPIC AND REGIONAL COUNCIL CHAIRS

Diversity, Equity, and Inclusion Council
Timothy Pinkston
Technology Policy Council
Virginia Dignum and Jeanna Matthews
ACM Europe Council
Rosa Badia
ACM India Council
Ponnurangam Kumaraguru
ACM China Council
Huadong Ma

PUBLICATIONS BOARD

Co-Chairs
Wendy Hall and Divesh Srivastava
Board Members
Rick Anderson; Tom Crick; Jack Davidson; Mike Heroux; Paola Inverardi; Odest Chadwicke Jenkins; Michael Kirkpatrick; Shan Lu; Mauro Pezzè; Rajeev Rastogi; Francesca Rossi; Bobby Schnabel; Stuart Taylor; Adelinde Uhrmacher; Philip Wadler; John West; Min Zhang; Tom Zimmerman

STAFF

DIRECTOR OF PUBLICATIONS
Scott E. Delman
cacm-publisher@cacm.acm.org

Executive Editor, ACM Magazines
Ralph Raiola
Senior Editor
John Stanik
Managing Editor
Thomas E. Lambert
Senior Editor/News
Lawrence M. Fisher
Web Editor
David Roman
Editorial Assistant
Sophia Navalta

Art Director
Andrij Borys
Associate Art Director
Margaret Gray
Assistant Art Director
Mia Angelica Balaquiot
Production Manager
Bernadette Shade
Intellectual Property Rights Coordinator
Barbara Ryan
Advertising Sales Account Manager
Ilia Rodriguez

Columnists
Saurabh Bagchi; Michael L. Best; Michael A. Cusumano; Peter J. Denning; Thomas Haigh; Leah Hoffmann; Mari Sako; Pamela Samuelson; Marshall Van Alstyne

CONTACT POINTS

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

REGIONAL SPECIAL SECTIONS

Co-Chairs
Virgilio Almeida, Haibo Chen, Jakob Rehof, and P J Narayanan
Board Members
Sherif G. Aly; Panagiota Fatourou; Chris Hankin; Sue Moon; Tao Xie; Kenjiro Taura

WEBSITE

https://cacm.acm.org

WEB BOARD

Chair
James Landay
Board Members
Marti Hearst; Jason I. Hong; Wendy E. MacKay

AUTHOR GUIDELINES

https://cacm.acm.org/author-guidelines/

COMPUTER SCIENCE TEACHERS ASSOCIATION

Jake Baskin
Executive Director

EDITORIAL BOARD

EDITOR-IN-CHIEF
James Larus
eic@cacm.acm.org

SENIOR EDITORS
Andrew A. Chien
Moshe Y. Vardi

EDITORS, IN MEMORIAM SECTION
Simson L. Garfinkel
Eugene H. Spafford

NEWS
Board Members
Lance Fortnow; Mei Kobayashi; Rajeev Rastogi; Vinoba Vinayagamoorthy

OPINION
Chair
Vivek Sarkar
Board Members
Roland Aydin; Saurabh Bagchi; Earl Barr; Andrew Begel; Mike Best; Judith Bishop; Marsha Chechik; Florence M. Chee; Danish Contractor; Lorrie Cranor; Janice Cuny; Jeremy Epstein; Armando Fox; James Grimmelmann; Mark D. Hill; Brittany Johnson; Bran Knowles; Raula Gaikovina Kula; Tim Menzies; Anne-Cécile Orgerie; Alessandra Raffaetà; Francesca Rossi; Eve Schooler; R. Benjamin Shapiro; Len Shustek; Bernd Stahl; Stuart Taylor; Loren Terveen; Marshall Van Alstyne; Matt Wang; Robert West

PRACTICE
Co-Chairs
Terence Kelly and Victor Yodaiken
Board Members
Satish Chandra; Vincent Hellendoorn; Mike Hoye; Pankaj Jalote; Saurabh Kadekodi; Donald Kossmann; Vinay Kulkarni; Lwin Khin Shar; Rui Maranhão; Stan Park; Steve Schirripa; Arie van Deursen

RESEARCH AND ADVANCES
Co-Chairs
m.c. schraefel and David Lo
Board Members
Alan Bundy; Peter Buneman; Haibo Chen; Monojit Choudhury; Jane Cleland-Huang; Gerardo Con Diaz; Anna Cox; Kathi Fisler; Nate Foster; Rebecca Isaacs; Trent Jaeger; Fabio Kon; Ben C. Lee; David Lo; Renée Miller; Ankur Moitra; Sarah Morris; Abhik Roychoudhury; Katie A. Siek; Daniel Susser; Charles Sutton; Thomas Zimmermann

RESEARCH HIGHLIGHTS
Co-Chairs
Shriram Krishnamurthi and Caroline Appert
Board Members
Martin Abadi; Sanjeev Arora; Maria-Florina Balcan; Selcuk Candan; Stuart K. Card; Jon Crowcroft; Lieven Eeckhout; Gernot Heiser; Takeo Igarashi; Nicole Immorlica; Natalie Enright Jerger; Srinivasan Keshav; Sven Koenig; Karen Liu; Claire Mathieu; Joanna McGrenere; Tamer Özsu; Tim Roughgarden; Chirag Shah; Guy Steele, Jr.; Wang-Chiew Tan; Robert Williamson; Andreas Zeller

Association for Computing Machinery (ACM)
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 869-7440; F (212) 869-0481

ACM Copyright Notice
Copyright © 2026 by Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or fee. Request permission to publish from permissions@hq.acm.org or fax (212) 869-0481.

For other copying of articles that carry a code at the bottom of the first or last page or screen display, copying is permitted provided that the per-copy fee indicated in the code is paid through the Copyright Clearance Center; www.copyright.com.

Subscriptions
An annual subscription cost is included in ACM member dues of $99 ($40 of which is allocated to a subscription to *Communications*); for students, cost is included in $42 dues ($20 of which is allocated to a *Communications* subscription). A nonmember annual subscription is $269.

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

ACM ADVERTISING DEPARTMENT

1601 Broadway, 10th Floor
New York, NY 10019-7434 USA
T (212) 626-0686
F (212) 869-0481

Advertising Sales Account Manager
Ilia Rodriguez
ilia.rodriguez@hq.acm.org

Media Kit acmmediasales@acm.org

COMMUNICATIONS OF THE ACM

(ISSN 0001-0782) is published monthly by ACM Media, 1601 Broadway, 10th Floor New York, NY 10019-7434 USA. Periodicals postage paid at New York, NY 10001, and other mailing offices.

POSTMASTER

Please send address changes to *Communications of the ACM*
1601 Broadway, 10th Floor
New York, NY 10019-7434 USA

Printed in the USA.

Association for Computing Machinery



One of my priorities is to make sure ACM remains at the technical forefront of computing.

DOI:10.1145/3847610

Elisa Bertino

Computing Is Changing; How Should ACM Change with It?

I have been an ACM member for more than 40 years. During this time, ACM has given me opportunities to learn, to meet great colleagues from around the world, to present and publish research, and, perhaps most importantly, to be part of a community. I am deeply grateful to ACM; serving as its President gives me an opportunity to give something back to an organization that has been an important part of my professional life.

As I begin my term as President, I have been thinking about a fundamental question: *Computing is changing—how should ACM change with it?*

Let's consider how computing has evolved. We moved from algorithms and programming to networked and distributed systems, and then to cloud computing, mobile computing, and data-intensive systems. Today, AI is changing how we develop hardware and software, secure systems, manage data, conduct research, and interact with computing systems. AI-based systems increasingly sense, reason, communicate, and act in the physical world. Quantum technologies may require our community to revisit some of the foundations of today's computing infrastructure. Computing is also moving into new environments, from autonomous vehicles and robots to UAVs and space systems.

This evolution leads to another fundamental question: *What is computing today?*

I believe ACM should help our community and more broadly our societies to answer this question, and, more importantly, help shape what computing will become.

One of my priorities is to make sure ACM remains at the technical forefront of computing. We have already started several initiatives in areas I believe to be critical today and in which ACM should have a key presence. These include a new conference on agentic AI and the ACM AI Leadership Summit—which we would like to evolve into a continuing initiative rather than a single annual event, as well as plans for activities in physical AI, including embodied AI.

In the area of quantum technologies, I have co-founded the ACM Conference on Post-Quantum and Quantum-based Security (ACM PQQS)—ACM's first quantum-related conference—and I believe we should explore a broader technical community and additional activities in this area.

Robotics and autonomous air and space computing are other areas in which many important computing challenges are emerging. These areas also show a trend that today is more important than ever. Interesting problems no longer fit neatly within a single computing discipline. Autonomous air and space systems, for example, require AI, networking, distributed systems, security, real-time computing, software engineering, data management, and human-computer interaction. Physical AI similarly brings together AI, robotics, systems, sensing, and interaction. ACM has communities with expertise in all these areas. One of our great opportunities is to bring them together.

Another important priority is the ACM Digital Library (DL). The DL is an extraordinary repository of com-

ACM Transactions on
Computing for Healthcare

ACM Transactions on Computing for Healthcare (HEALTH) is a multi-disciplinary journal for the publication of high-quality original research papers, survey papers, and challenge papers that have scientific and technological results pertaining to how computing is improving healthcare.



For further information and to submit your manuscript, visit health.acm.org

puting knowledge, but I believe it can become much more. In particular, we should ask how it can better serve practitioners and educators in addition to researchers. AI provides opportunities for new services that can help people discover, connect, understand, and use computing knowledge. We should explore these opportunities responsibly while preserving the quality and trust associated with ACM content. We also need to think about the long-term sustainability of the DL, including new partnerships and a funding campaign to support its future development.

A third priority is open research in computing. Openness means more than making papers accessible. Computing research increasingly depends on datasets, benchmarks, software, models, and experimental infrastructure. We should explore creating a journal dedicated to datasets and benchmarks across computing—not only in AI. At the same time, AI poses particularly difficult questions for experimental research. Much leading research—both on AI itself and on computing systems that incorporate AI—is conducted in industry using proprietary models, datasets, and computational infrastructure that cannot readily be made available to the broader research community. How should such work be evaluated? What should reproducibility mean under these conditions? What information and evidence must researchers provide so that their results can be independently and meaningfully assessed?

I believe ACM can bring academia and industry together to develop new methods and standards for rigorous experimental research in the AI era. This effort should also address access to AI models and other essential resources, and how greater openness can be reconciled with legitimate concerns about security, intellectual property, cost, and responsible use.

To pursue initiatives such as these, ACM itself must be able to move quickly. We need mechanisms that make it easier for our communities to create conferences, summits, and visioning workshops around emerging topics. We should also consider organizational structures that allow ACM to focus on important multidisciplinary areas that cross the boundaries of existing SIGs. This does not mean replacing our existing communities. On the contrary, it means making it easier for their expertise to come together when new challenges require it.

The Digital Library is an extraordinary repository of computing knowledge, but I believe it can become much more. In particular, we should ask how it can better serve practitioners and educators in addition to researchers.

None of this can be accomplished by a President—or even by the ACM Council—alone. ACM's greatest strength has always been its community and its volunteers. During my more than 40 years as a member, I have seen what ACM volunteers can accomplish when people with different expertise, experiences, and perspectives come together around an idea. My goal for the next two years is therefore not simply to launch a collection of new initiatives. It is to help ACM become even better at connecting people, disciplines, and ideas.

I look forward to listening to your ideas and working with all of you. Together, I hope we can help ACM continue to serve our community and contribute to shaping the future of computing. C

Elisa Bertino (bertino@cs.purdue.edu) is president of the Association for Computing Machinery and the Samuel D. Conte Distinguished Professor of Computer Science at Purdue University, West Lafayette, IN, USA.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

© 2026 Copyright held by the owner/author(s).

If we already possess increasingly sophisticated methods for detecting hazards, why have we invested comparatively little effort in communicating their human context?

DOI:10.1145/3844133

Vinton G. Cerf and Harsh Verma

Why We Need Occupancy Intelligence during Emergencies

Today, billions of connected devices continuously observe our world. Satellites monitor storms from space. Seismic networks detect earthquakes within seconds. River gauges report rising floodwaters. Weather radar tracks tornadoes. Wildfire sensors identify rapidly changing fire conditions. Emergency alerts can now reach millions of people almost instantly through mobile phones, television, radio, and the Internet.

These advances represent remarkable achievements in engineering and global collaboration. We have become increasingly successful at answering questions such as: What is the Emergency? Where did it happen? How severe is it? When will it occur?

Other important questions, however, often remain unanswered: *Where are the people and how many may need help?*

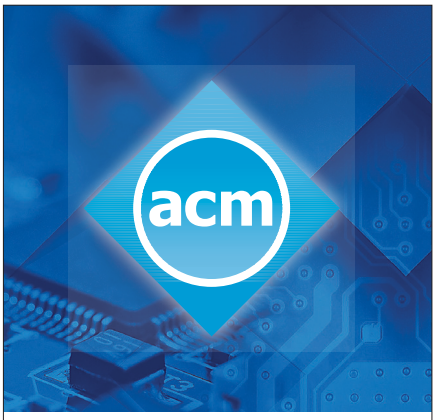
As technology has evolved over the years, our modern digital systems, including emergency alerting systems, have been extraordinarily successful in describing hazards. The standards for alerting do not describe the dynamic human environment surrounding those hazards. That information is critically important during emergencies.

A magnitude 7.0 earthquake affecting a downtown business district at 3:00 a.m. on a Tuesday presents a very different operational picture one occurring at 3:00 p.m. on a Friday afternoon. A hurricane approaching a coastal city during a holiday weekend differs significantly from one arriving after a successful evacuation. The hazard itself may be identical, but the number of people exposed, their locations, and their ability to respond can vary dramatically.

Emergency managers make decisions based not only on the characteristics of the hazard but also on the characteristics of the affected population. Which hospitals may soon receive large numbers of casualties? Which transit stations are crowded? Which schools are occupied? Which evacuation routes are already congested? Which shelters still have available capacity?

During a U.S. Geological Survey Pilot Licensing Agreement pilot, Harsh Verma and his team were investigating how real-time crowding and occupancy modeling could complement Earthquake Early Warning systems. The objective was not to improve the hazard models themselves but to better understand their operational human impact. The exercise revealed that although modern alerting systems have become remarkably effective at characterizing hazards, they generally lack standardized ways to communicate information about the people who may be affected. If we already possess increasingly sophisticated methods for detecting hazards, why have we invested comparatively little effort in communicating their human context?

Hazard detection and warning systems evolved to measure physical events. Population information evolved within entirely different systems—transportation networks, building management systems, mobile devices, public safety operations,



Advertise with ACM!

Reach the innovators and thought leaders working at the cutting edge of computing and information technology through ACM's magazines, websites and newsletters.

◇◆◇◆◇

Request a media kit with specifications and pricing:

Ilia Rodriguez
+1 212-626-0686
acmmediasales@acm.org



and countless other sources. Each serves its own purpose. Rarely were they designed to work together during rapidly evolving emergencies.

Ironically, the Internet facilitates the connection of much of this information. Cities are becoming increasingly instrumented. Public transportation systems, smart buildings, infrastructure sensors, and mobile technologies continuously generate information that helps us understand how people move through the built environment. The challenge is no longer whether such information exists. The challenge is to determine how it can be represented, exchanged, and trusted in ways that are useful during emergencies while preserving privacy and public confidence.

Occupancy intelligence should never be confused with surveillance. Emergency responders rarely need to know *who* is present. More often, they need to understand whether a location contains dozens, hundreds, or thousands of people; whether occupancy is increasing or decreasing; and how reliable that information may be. Properly designed systems can provide operational awareness using aggregated, anonymized, and privacy-preserving information without identifying individuals.

As our digital infrastructure continues to evolve, perhaps the Internet and other communication systems will be able to carry more than descriptions of hazards. Perhaps they should also carry reliable contextual information that helps responders understand the potential human consequences of emergency situations and hazardous conditions.

Achieving this objective will require careful consideration of data quality, confidence, timeliness, governance, interoperability, and, above all, trust. These are not merely technical challenges. They are questions for computer scientists, emergency managers, standards organizations, policy makers, public agencies, and technology companies working together.

The Internet and its applications have continually evolved by solving problems earlier generations could not fully anticipate. We improved routing as networks expanded. We strengthened security as new threats emerged. We developed mechanisms for establishing trust in increasingly complex digital environments.

Properly designed systems can provide operational awareness using aggregated, anonymized, and privacy-preserving information without identifying individuals.

Perhaps the next evolution is ensuring the information we move provides the context needed to use it to make better decisions. Knowing where the earthquake will strike is important. Knowing where people are when it strikes may help save lives.

If occupancy intelligence can become an important component of emergency decision making, how should it be represented, exchanged, and trusted? Those questions are now beginning to be explored by researchers and standards communities alike. International emergency-alerting standards are evolving to incorporate occupancy intelligence while preserving interoperability, privacy, and public trust. It seems to us that this is an area worthy of increased attention with the potential to offer significant life-saving returns. C

Vinton G. Cerf (vgcerf@gmail.com) is CEO of Tolero Networks. He served as ACM president from 2012–2014..

Harsh Verma (harsh@peoplesense.ai) is founder and CEO of Glocol Networks and PeopleSense Digital Corp., and chair of the ACM Sacramento Chapter.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3822413

From Absence to Advocacy: Shaping the Future for Women in Tech



NAME
Oluwasefunmi 'Tale Arogundade
BACKGROUND
Born and raised in Are Ekiti, Ekiti State, Nigeria; now living in Abeokuta Ogun State, Nigeria.
CURRENT JOB TITLE/EMPLOYER
Pioneer Dean of College of Computing Sciences; Prof. at the Dept. of Software Engineering and Information Systems, Federal Univ. of Agriculture Abeokuta, Nigeria.
EDUCATION
Ph.D. in Computer Software and Theory, Graduate Univ. Chinese Academy of Science

My career path has not been an easy undertaking but one motivated by aspiration and ambition. I noticed a gap and pursued a dream to bridge it, leading to a results-oriented, fulfilling career.

My interest in computer science dates back to my undergraduate years. I was drawn to study the discipline because of its intellectual depth, capacity for innovation, and value to life in addressing real-world complex problems. After graduating as the top student in my master's cohort, I was appointed as an assistant lecturer. Funny enough, throughout my undergraduate and postgraduate studies, I did not come across a single female academic instructor in my department, which became strong, unbendable motivation to be part of those changing the narrative and made this ambition challenging but necessary.

Soon after my appointment, I received the TWOWS postgraduate fellowship to pursue a Ph.D. at the Graduate University of the Chinese Academy of Sciences. At the time, I was married with two young children. Accepting the fellowship meant leaving my family in Nigeria, a difficult and deeply personal sacrifice. However, I understood that advancing my academic qualifications was essential, not only for my career progression but also for sustaining my place within academia. As policies evolved in my home country, a doctorate became a mandatory qualification for retention of academic position and advancement. Fortunately, I had already begun my doctoral studies. The experience expanded my global academic exposure, solidified my identity as a researcher, and positioned me for greater contributions to the field.

Beyond these institutional expectations, my gender-based challenges persist. There remains a pervasive belief in some quarters that women may be unable to meet the demands of academics, particularly in balancing professional and family responsibilities. Navigating through all these took deliberate effort, dedication, persistence, and personal discipline. Also by my side is the unwavering support of my family, especially my spouse, whose encouragement has been instrumental throughout my journey. Over the years, I have continued to build my academic profile through research, teaching, and leadership. Today, as a professor of software engineering, Pioneer Dean of the College of Computing Sciences, and leader of the SmartAgro research cluster, I contribute to advancing knowledge at the intersection of computing and agriculture, an area with significant impact, particularly in the developing world. NimTrack was the first hardware innovation devised by my research team. It is a network of sensors that assists cattle owners in locating and monitoring the health status of their animals thereby preventing animal rustling and mortality respectively.

My career has also been graced by numerous fellowships and international opportunities, including the OWSD postgraduate fellowship, the Intl. Federation of University Women (IFUW) award, Heidelberg Laureate Forum (HLF) participation, and a TWAS-DFG postdoctoral fellowship in Germany. These experiences have broadened my perspective and strengthened my commitment to global collaboration.

Equally important is my role in supporting the next generation of women in computing. As chair of the ACM-W Ibadan Nigeria Chapter, I promote inclusion, mentorship, and visibility for women in the field. I remain committed to ensuring more women see computing as a viable and rewarding career.

Reflecting on my journey, what began as a response to absence has evolved into sustained effort toward advocacy and impact. Though the road remains challenging, it has been deeply fulfilling. My career navigation has been about creating opportunities for those following me. C

This work is licensed under a Creative Commons Attribution International 4.0 License.
© 2026 Copyright held by the owner/author(s).

BLOG@CACM

In each issue of *Communications*, we publish selected posts or excerpts from the many blogs on our website. The views expressed by bloggers are their own and not necessarily held by *Communications* or the Association for Computing Machinery.

Read more blogs and join the discussion at https://cacm.acm.org/blog.

Converting Text to SQL, and Computing to Fix Red Tape

Michael Stonebraker and Peter Baile Chen address issues with data warehouses, while Vinay K. Chaudhri and Patrick A. McLaughlin consider AI's role in regulations.



MICHAEL STONEBRAKER AND PETER BAILE CHEN

If You Think You Can Do Real-World Text-to-SQL

DOI:10.1145/3838721
https://bit.ly/45JJqeU

Data warehouses are often subject to a preponderance of ad hoc queries. They came into existence in the 1990s to support historical data stores of customer-facing data, especially sales data. At that time, it was widely reported that they paid for themselves based on better stock rotation in less than six months. For example, if pet rocks are "out" and Barbie dolls are "in," then discount the pet rocks and move them up front; tie up the manufacturer of Barbie dolls with a big order so competitors cannot get merchandise.

Although there are repeated queries (for example, "produce the weekly sales report"), much of the traffic is ad hoc in nature, including the above example. Historically, data warehouses were fronted by business intelligence tools such as Business Objects, Power BI, and Tableau. This allowed many queries to be coded by business users using a simple graphical interface. Obviously, one would like to query data warehouses and other large, structured data stores in natural language to further empower non-programmers. It would have the added benefit of not requiring familiarity with a bespoke user interface. With the advent of large language models (LLMs) and agentic AI, this becomes a real possibility, and there are public benchmarks with leaderboards that show promising results.

On Spider 1.0,[a] the leaderboard had accuracy in the 80+% range, while on Bird-SQL,[b] accuracy was above 90%. However, both benchmarks have the following issues:

▸ **The data is in "the pile," so LLMs can train on it.** It is widely known that an LLM can only find data it has seen before. Almost all data warehouses we know about are behind serious enterprise access controls and are not publicly available.

a https://yale-lily.github.io/spider
b https://bird-bench.github.io/

▸ **Real data warehouses exhibit "schema rot."** Inevitably, the warehouse begins life with a well-structured schema. However, the schema of a typical warehouse is updated as business conditions change, the company buys or sells business units, and regulations change. Often schema changes occur multiple times a year. It would be wonderful if enterprises spent the time to construct a well-structured replacement schema. Invariably this is not the case, and schemas "rot" over time. The result after a few such changes is non-intuitive table and column names. For example, there may be six different columns in various tables labeled "salary." Such columns may have overlapping non-documented semantics; one salary column may be for "net after taxes and commissions," while another one may be for "gross salary." Materialized views may be used to speed up popular queries. These characteristics give an LLM multiple ways to solve a query and to generate headaches.

▸ **Real data warehouses often have idiosyncratic data.** For example, in the MIT administration data warehouse containing information on students, faculty, building, classes, and so on,

there are mentions of "J-term" (a one-month term in January between fall and spring semesters) and building numbers (MIT buildings are numbered, not named). Such bespoke information is difficult for an LLM.

▸ **Real data warehouse queries are more complex than the student-generated ones in Spider and Bird-SQL.** They rarely have just one join and often two or three.

Spider 2.0[c] has addressed the last issue, but not the first three. The leaderboard of agentic-AI solutions for this benchmark is in the 90+% range.

In our opinion, real data warehouses have all the issues mentioned above. Hence, any text-to-SQL benchmark should address the difficulties of real-world data stores. Those that do not are of academic interest only, and do not reflect reality.

Beaver Benchmark

For the last two years, we have been trying LLM-based text-to-SQL on real workloads from real data warehouses. We started with the above-mentioned MIT data warehouse, an Oracle data warehouse of 1,400+ tables that is queried by many units at MIT (including CSAIL). Typically, business users pose a question to IT personnel who translate the request to SQL, often with clarifying dialog between the two people. We got real queries from the Oracle log and generated the corresponding natural language with the assistance of real users. This generated a collection of (natural language, gold SQL) pairs. Later we applied the same methodology to three other data warehouses. With permission, we anonymized and extended the queries and the data to produce the Beaver benchmark.[d] One should clearly note the four issues noted above are all present in this benchmark. As such, we do not expect to get the same high accuracy as Spider/Bird. More details on the benchmark are published online.[e]

On this benchmark, a pure LLM generated an accuracy score of zero. Adding RAG, prompt engineering, and agentic AI raised accuracy to the 10+% range. See Jindal[f] for results on this sort of model. To get a higher score, we gave the LLM the specific tables in the *from* clause of the gold SQL and the actual *join* clauses when there were multiple tables. That generated accuracy in the 30+% range. A leaderboard for results on Beaver is available online.[g] Note that this is 50+% lower in accuracy that the public benchmarks and is the difference between "the technology has promise" and "the technology does not work."

We would welcome systems and ideas on how to make LLM/agentic AI technology work much better on real-world business applications, and we have made Beaver publicly available to assist in this process. So have at it! Our own ideas on how to do better are contained in the Rubicon system,[h] so take a look.

A simple conclusion can be drawn from this discussion. Good results using traditional LLM/agentic AI technology are likely only on data warehouses with clean schemas, simple queries, and little idiosyncratic data. Otherwise, try Rubicon or your good idea. C

c https://spider2-sql.github.io

Any text-to-SQL benchmark should address the difficulties of real-world data stores. Those that do not are of interest only, and do not reflect reality.

d https://beaverbench.github.io/
e https://beaverbench.github.io/#overview
f https://arxiv.org/pdf/2602.08320
g https://beaverbench.github.io/#leaderboard
h https://arxiv.org/abs/2604.21413

Michael Stonebraker is a professor of computer science at the Massachusetts Institute of Technology, where he specializes in database management. For his contributions, he was awarded the ACM A.M. Turing Award in 2014.

Peter Baile Chen is a Ph.D. student at the the Computer Science and Artificial Intelligence Laboratory of the Massachusetts Institute of Technology, where he works with Michael Stonebraker at the intersection of AI and data systems.

© 2026 Copyright held by the owner/author(s).



VINAY K. CHAUDHRI AND PATRICK A. MCLAUGHLIN
Computing Can Help Fix Red Tape, but It Shouldn't Write the Rules

DOI: 10.1145/3838734
https://bit.ly/4cj8v42

Regulatory accumulation imposes a gigantic hidden tax on the U.S. economy. AI and advanced computing offer new opportunities to help address this problem and modernize the regulatory system, but only if we intentionally build the right computational infrastructure—one allowing regulations to be more easily understood, efficiently executed, and continuously improved by human decision makers.

An influential estimate[a] (co-written by one of the authors of this post) suggested that if federal regulation had remained at its 1980 level, the economy would have been nearly 25% larger by 2012—equivalent to roughly $47,000 per household today. That startling figure was informed by projects such as QuantGov,[b] which analyzed millions of words of regulatory text. The next question is: How do we go beyond measuring regulatory complexity and help governments manage it?

A white paper[c] from the Industry Committee of the Computing Research Association (CRA-I) outlines how this might look, with advances in AI, knowledge representation, and cryptography modernizing the regulatory system.

These tools must be treated as infrastructure that supplements, rather than replaces, human judgment, and not as a call for automated lawmaking. Regulation embodies public choices about a mix of the tangible and intangible: risk, fairness, property rights, innovation, and institutional authority. The ultimate decisions, therefore, are fundamentally human and can-

a https://www.mercatus.org/research/working-papers/cumulative-cost-regulations
b https://www.quantgov.org/
c https://cra.org/industry/wp-content/uploads/sites/9/2026/07/Unsnarling-the-Red-Tape_-Computational-Infrastructure-for-Regulatory-Systems.pdf

Association for Computing Machinery

Career & Job Center

The #1 Career Destination to Find Computing Jobs.



Connecting you with top industry employers.

Check out these new features to help you find your next computing job.

- Access to new and exclusive career resources, articles, job searching tips and tools.
- Gain insights and detailed data on the computing industry, including salary, job outlook, 'day in the life' videos, education, and more with our new Career Insights.
- Receive the latest jobs delivered straight to your inbox with **new exclusive Job Flash™ emails**.
- Get a free resume review from an expert writer listing your strengths, weaknesses, and suggestions to give you the best chance of landing an interview.
- Receive an alert every time a job becomes available that matches your personal profile, skills, interests, and preferred location(s).

Visit https://jobs.acm.org/

not be made legitimate by computer analysis alone.

What modern computing can do, however, is groundbreaking. With more than one million commands on the books at the federal level alone, the regulatory corpus is too complex for unassisted human analysis. Computing cannot tell us which regulations are worth having, but it can contribute to a serious modernization agenda in three ways.

First, it can make regulations easier to understand and comply with. To do this, we need not turn natural language laws into computer code. Instead, agencies should publish structured representations alongside legal text so people and software can more easily identify who is being commanded to do what, under what conditions, subject to which exceptions, and according to which procedures.

Second, some parts of regulation are naturally suited to automated compliance. Reporting thresholds, eligibility tests, filing requirements, and certain permitting conditions are often clear enough that automation can lower compliance costs and reduce delays. But the boundary matters. Wherever rules require discretion, balancing, legal interpretation, or political judgment, software should assist rather than decide.

Properly designed automation can even strengthen privacy when designed around minimizing data sharing rather than centralized surveillance. Cryptographic techniques such as zero-knowledge proofs can allow firms or individuals to demonstrate that they satisfy certain requirements without exposing all their underlying data.

Third, automated tools can flag problematic regulations and reveal where excessive, expensive complexity accumulates. In doing so, they can identify two particularly problematic types of rules: those that serve legitimate public purposes but are unnecessarily expensive to comply with; and those that are obsolete, duplicative, contradictory, or more costly than beneficial. Tools should not dictate reform, but they can provide auditable evidence for human review.

In short, imagine a new way of publishing regulations in which structured data and machine-readable rules are made available alongside legal text. Developers could build compliance tools from them. Agencies could automatically flag potential problems for expert review. Researchers could measure how regulations evolve over time. Policymakers could see, not guess, where complexity is accumulating.

Computing cannot tell us which regulations are worth having, but it can contribute to a serious modernization agenda.

Computational methods alone are insufficient. Their benefits depend on redesigning administrative processes to take advantage. In addition to publishing automation-friendly versions of regulations, this means streamlining approvals, adopting standardized digital workflows, and increasing transparency. Just as important, agencies should build expert-validation workflows so that computational systems generate reviewable recommendations rather than unaccountable decisions.

For decades we have treated regulation as a growing collection of legal documents. We should instead treat it as a living system—one that can be measured, analyzed, and continuously improved. Machines make that possible, helping us see what our laws actually require, where they conflict, how they accumulate, and what they cost. In a modern democratic system, that is the proper role of technology: not to replace human judgment, but to discipline it with better information. C

Vinay K. Chaudhri is a principal scientist at Knowledge Systems Research LLC, where he currently supports a National Science Foundation initiative on Knowledge Axiomatization.

Patrick A. McLaughlin is a research fellow at the Hoover Institution, a visiting research fellow at the Pacific Legal Foundation, and an affiliated member of the Frankfurt Competence Center for German and Global Regulation.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3842411

Logan Kugler

The End of the Coder?

As AI agents transition from simple coding assistants to end-to-end engineers, the profession is shifting from writing syntax to auditing complex systems.

IN JANUARY 2026, Anthropic CEO Dario Amodei made a prediction that sent ripples through the technology sector: The world might be only six to 12 months away from AI models capable of performing all software engineering tasks end-to-end. The timeline was aggressive, but the evidence was already mounting within the walls of the major AI labs themselves. Shortly after Amodei's statement, Boris Cherny, the head of Claude Code at Anthropic, admitted that 100% of his own code is now AI-generated.

For decades, the career of a "software developer" was defined by the ability to translate human requirements into precise, machine-readable syntax. Yet a rapid succession of AI releases in early 2026 has brought that definition to a breaking point.

On Feb. 2, 2026, OpenAI launched the Codex desktop app—a tool designed to create not just simple code snippets, but also long-running, multi-agent workflows involving isolated worktrees and automated deployment. Weeks later, GPT-5.3-Codex arrived, shifting the industry's central question from "Can the model write code?" to "Can humans direct and audit a fleet of agents fast enough?"

The impact on individual practitioners was immediate.



IMAGE BY ANDRIJ BORYS ASSOCIATES WITH ASSISTANCE OF SHUTTERSTOCK GEN AI

"I am no longer needed for the actual technical work of my job," said Matt Shumer, co-founder of Otherside AI, in an essay on X[a] in early 2026. Shumer described a workflow common among advanced AI users:

"I describe what I want built, in plain English, and it just ... appears. Not a rough draft I need to fix; the finished thing. I tell the AI what I want, walk away from my computer for four hours, and come back to find the work done; done well, done better than I would have done it myself, with no corrections needed."

If AI can write, debug, and deploy software for near-zero cost, what is

a https://x.com/mattshumer_/status/2021256989876109403



A great speaker can make the difference between a good event and a WOW event!

Take advantage of ACM's Distinguished Speakers Program to invite renowned thought leaders in academia, industry, and government to deliver compelling and insightful talks on the most important topics in computing and IT today. ACM will cover the cost of transportation for the speaker to travel to your event.

speakers.acm.org

Association for Computing Machinery

the impact on human software development professionals?

From Implementation to Strategic Oversight

The consensus among authorities is that while the "coder" may be endangered, the "engineer" remains essential, though that role is being redefined as part product manager, part reviewer, and part AI orchestrator.

James Ivers, lead of the AI Workflows and Architecture Modernization group at Carnegie Mellon University's Software Engineering Institute, argues that the value proposition of a professional engineer has always extended far beyond syntax.

"Coding is often the easy part," Ivers said. "Great software engineers impact much more of the software development life cycle than just slinging code. Requirements analysis, architecture and design, effective testing strategies, planning, and stakeholder management—all of these are critical activities for project success."

Ivers suggested a mental model that distinguishes between "coders" and "software engineers." In this view, coders are language experts who operate within bounds defined by others, such as Jira tickets or specific designs. Software engineers, conversely, operate in ambiguous spaces to discover and create those bounds, engaging with stakeholders to determine requirements and priorities.

"Within this mental model, coders are much more likely to be impacted or even displaced by AI," Ivers said.

Bill Nichols, lead of the Applied Measurement and Experimentation Initiative at Carnegie Mellon University, said that the differentiator is moving up the abstraction stack.

"The value proposition shifts from being a scarce source of code to being a scarce source of well-formed decisions," Nichols said. As implementation costs drop toward zero, he said, the work shifts toward understanding domains, making trade-offs explicit, and ensuring systems behave as intended under real-world constraints.

The Quality and Security Bottleneck

While AI can generate code at a scale that outpaces a human's, it remains an imperfect actor. Ivers views software engineering as an evolution of practices designed to compensate for human error—finding mistakes as early as possible to reduce the cost of fixing them.

"None of these motivations go away when using AI," he said. "We are replacing one imperfect actor with another. AI is imperfect and there will be mistakes."

Software engineers are uniquely positioned to bring discipline to these projects by anticipating where errors can be introduced, spotting them in practice, and guiding the AI toward intended solutions.

Amy J. Ko, a professor and Associate Dean for Academics of The Information School at the University of Washington, said there is no evidence agentic AI can devise good plans without expert human review and well-designed architectures.

"Expertise has always been marked by a deep knowledge of software qualities and how they are achieved through implementation; understanding architectural complexity; capacity to continuously learn and change practices; providing credible, honest, trustworthy information, and a long tail of other soft skills," Ko said.

This reliance on human judgment creates a new set of bottlenecks. Nichols said the limiting factor in the software lifecycle is moving from "Can we build it?" to "Can we understand, test, and justify it?" He draws a sharp distinction between verification and validation: Verification (checking if a system conforms to specs) can be automated, but validation (deciding if the system provides the right capabilities for its environment) remains a judgment-heavy human process.

Without this oversight, agent-generated codebases risk turning into unmaintainable sludge, Ko said.

The Broken Junior Pipeline

However, there is concern for some professional roles; perhaps the most urgent relates to junior developers. Historically, entry-level roles allowed developers to build architectural proficiency through the manual trial and error of coding. With AI agents now capable of doing the work typically as-

signed to juniors, that foundational practice is under threat.

"We have no way of teaching, training, or educating software developers to be senior or architect-level software engineers" without that foundational experience, Ko warned. She said industry currently relies on existing senior talent while "deferring investing in future senior developers to instead purchase compute," a strategy she called unsustainable. Aside from years of practice designing and building software in teams, there are no known pedagogies or research that can produce that level of high-tier skill, Ko said.

A New Pedagogy of Feedback

Despite these structural hurdles, some see the potential for a new kind of learning. Nichols said AI-assisted development could accelerate system-level understanding, not hinder it.

"Manual coding has been one path to developing intuition, but it's limited," Nichols explained. "Feedback is often delayed and indirect, and design decisions may reveal their consequences only much later. Moreover, coding is local behavior; architecture is system behavior."

When an engineer specifies behavior and an agent produces an implementation, the feedback loop shifts from "Did I write this correctly?" to "Did I specify this correctly?" This provides immediate feedback on whether the specification was clear and sufficient, allowing junior engineers to focus on aspects of the system that matter most for long-term behavior.

Ivers echoed the "think beyond coding" approach, suggesting that problem decomposition and domain knowledge will be paramount for future-proofed developers.

"AI will need guidance. It also needs monitoring, which means verification and validation skills are important, and refinement, so architecting new agent workflows or improving management of new resources like tokens are huge here too," Ivers said.

The Jevons Paradox of Code

The economic outlook for the profession remains a subject of intense debate at present. While implementation work is being commoditized, Nichols pointed to Jevons paradox,[b] an economic theory in which efficiency gains in a resource (such as coal or, in this case, code) lead to an increase in its total consumption.

The consensus among authorities is that while the "coder" may be endangered, the "engineer" remains essential.

"As the cost of implementation falls, more software tends to be built, especially lower-stakes systems that were previously uneconomical," Nichols said. This could expand the overall scope of software work and create new entry points for early-career engineers, helping to address concerns about an empty entry-level talent pipeline.

However, the future path is more of a bend than a straight continuation. The professional's role is evolving from progressively better implementers toward roles focused on shaping and controlling system behavior under real-world constraints. Hiring is expected to tilt toward skills in requirements, system design, risk management, and the ability to work effectively with AI tools as collaborators.

As today's frontier models and their successors take over the keyboard, the human engineer's most important tool is no longer the integrated development environment; it is their judgment. C

b https://news.northeastern.edu/2025/02/07/jevons-paradox-ai-future/

Logan Kugler is a technology writer specializing in artificial intelligence based in Tampa, FL, USA. He has been a regular contributor to *Communications* for 15 years and has written for nearly 100 major publications.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 ACM 0001-0782/26/10

ACM Member News

DESIGNING PROTOCOLS FOR FEDERATED LEARNING



Xiaohua Jia is Chair Professor in the Department of Computer Science at City University of Hong Kong.

Jia earned undergraduate and master's degrees in computer science from the University of Science and Technology of China, and a Ph.D. in information science from the University of Tokyo.

After obtaining his Ph.D. in 1991, Jia joined the faculty at the University of Queensland in Australia, and in 1996 he left for City University of Hong Kong, where he is today.

Jia's research is focused on data privacy and security in AI and machine learning, with a particular emphasis on federated learning.

Federated learning enables collaborative model training without moving data from its local source, essentially moving the model to the data. This approach introduces its own challenges, such as how to verify participants genuinely possess the data they claim, and that the training was computed locally.

"As modern AI systems rely on vast amounts of data, much of the most valuable information remains siloed within private organizations such as hospitals and financial institutions, where sharing raw data is impractical or prohibited," Jia said.

"My research addresses these issues through the development of protocols for 'proof of data' and 'proof of training,' allowing federated learning systems to validate both data ownership and computational integrity."

Looking ahead, Jia envisions open, privacy preserving federated learning platforms enabling global collaboration across institutions and regions. Such platforms, he said, could support applications from medical research to financial modeling while ensuring sensitive data remains secure, local, and verifiably computed.

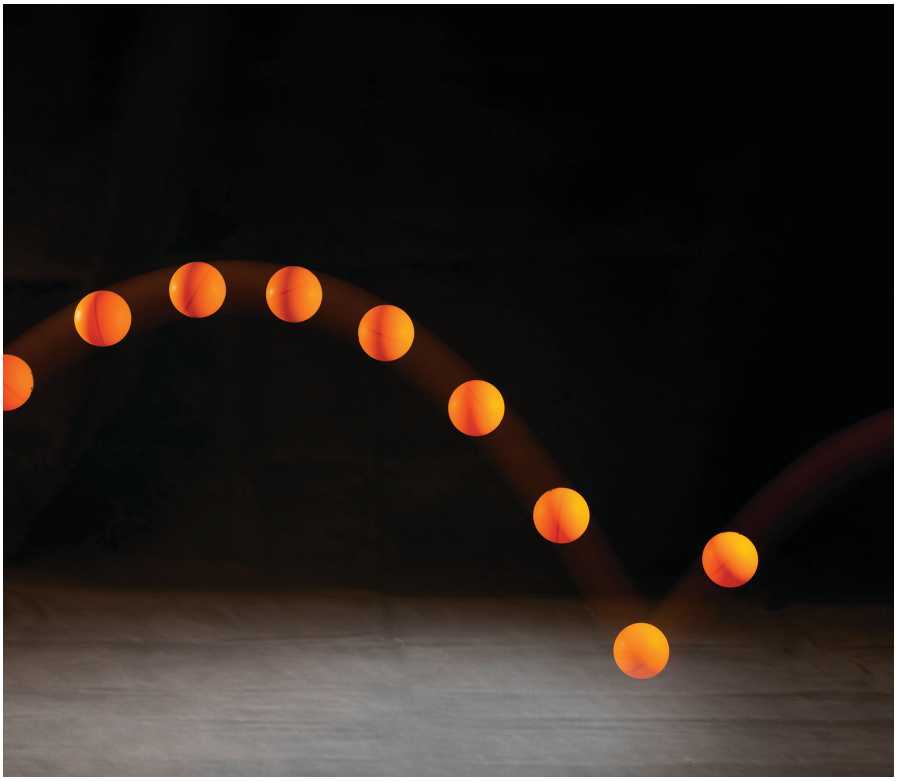
—John Delaney

DOI:10.1145/3842412

Samuel Greengard

Beyond LLMs: A Post-Transformer World Emerges

A focus of the next decade might be on teaching AI how to master reality and understand how things work in the physical world.



The rapid ascent of large language models (LLMs)—and their growing role in everyday life—masks a fundamental problem: Generative pre-trained transformer (GPT) models hallucinate, struggle with memory-intensive tasks, consume enormous compute and energy resources, and at times behave unpredictably.

Researchers have a name for the resulting unevenness: jagged intelligence. Systems solve graduate-level problems and then stumble over something a child would get right. What's more, failures often arrive without any warning.

This makes LLMs valuable for many tasks, but it renders them unreliable for predicting basic physical actions, such as how a ball bounces or a cat jumps. A popular test makes the point: "I put a coin inside a coffee mug. I carried the mug into the kitchen and turned it upside down on the counter. Then I walked back to the living room with the mug. Where is the coin?" Until recently, models routinely answered that the coin was in the living room, still inside the mug. They couldn't simulate the coin falling out. Today's leading models answer it correctly—but a correct answer to a widely circulated puzzle is not the same as intrinsic knowledge, or even a basic understanding of physics. Vary the puzzle slightly and the failure returns.

LLMs are valuable for many tasks, but are unreliable for predicting basic physical actions such as how a ball bounces or a cat jumps.

As a result, researchers are exploring post-transformer[a] AI frameworks. Rather than simply building bigger LLMs, the idea is to infuse models with a better understanding of spatial relationships, improve their memory, and incorporate new types of reasoning.

Concepts include "world models"[b] that simulate how people, objects, and environments behave; agentic architectures that break down problems into smaller components and use multi-step reasoning; selective state space models (SSMs)[c] capable of long-sequence modeling with linear scaling; and biologically inspired neural memory modules[d] that operate more like the human brain.

Said Yann LeCun,[e] Jacob T. Schwartz Professor of Computer Science at New York University and a 2018 ACM Turing Award laureate: "We are never going to achieve human-level intelligence in AI simply by training text. The world is more complicated than language."

Model Behaviors

If the last decade of AI has focused heavily on teaching computers to

a https://www.ibm.com/think/topics/transformer-model
b https://www.nvidia.com/en-us/glossary/world-models/
c https://www.ibm.com/think/topics/state-space-model
d https://aipapersacademy.com/titans/
e http://yann.lecun.com/

IMAGE FROM SHUTTERSTOCK.COM

master language, the next decade might focus on teaching AI how to master reality and understand how things work in the physical world. Although large language models aren't likely to disappear, the AI universe is likely to become more diverse.

"Today's LLM models are a bit more complex than similar neural networks from 1991, but not by a lot," said Jürgen Schmidhuber,[f] co-chair of the GenAI Center at KAUST, Scientific Director at Swiss AI lab IDSIA and a pioneer[g] in the field of recurrent neural networks (RNNs). "Passing the 'Turing Test'[h] is much easier than [developing] true AI in the physical world. Predicting the next token by itself is not enough to produce artificial general intelligence (AGI)[i]."

Much of this trouble traces back to memory, said Phillip Isola,[j] associate professor of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology. "Transformers don't natively have memory in the sense of how we think about it in humans," he said. The problem with LLMs is rooted in the way they process information. As their context window shifts forward, they overwrite tokens naively, he added. This causes errors and hallucinations, but it also contributes to models losing their train of thought as they tackle complex tasks.

While the ability to tamp down hallucinations has improved thanks to guardrails, augmented retrieval, and reasoning-oriented training, the underlying LLM engine has not evolved. Models still generate plausible, even convincing output—sometimes right and sometimes wrong—without any understanding of how the world actually works. They are far less likely to invent a citation, a statistic, or even a news event, but they still cannot reason about what would happen if a ball rolled into the street in front of an autonomous vehicle. Is it only a ball? Is there a child chasing it?

f https://people.idsia.ch/~juergen/
g https://people.idsia.ch/~juergen/who-invented-transformer-neural-networks.html
h https://www.britannica.com/technology/Turing-test
i https://www.ibm.com/think/topics/artificial-general-intelligence
j https://web.mit.edu/phillipi/

The problem with LLMs is rooted in the way they process information. As their context window shifts forward, they overwrite tokens naively.

Transformer models should ideally serve as just one layer in the overall AI stack, Isola said. Yet, more layers increase the odds that something can go astray. While LLMs appear fluent, they often struggle with facts, tracking goals, long-term planning, and adapting to changes. "Transformer models have many limitations, including a lack of understanding of spatial relationships that are a critical part of our world," added Aditi Raghunathan,[k] assistant professor in the Computer Science Department at Carnegie Mellon University.

Seeking Transformation

Impressive language skills notwithstanding, transformer-based models function more like enormous databases than sources of intelligence, Schmidhuber said. Transformer models excel at "fuzzy" tasks like writing or brainstorming ideas, but they are ill-suited for scientific calculations or exact, step-by-step reasoning, such as reconciling financial records.

The gap is not confined to the physical world; it shows up in formal reasoning too. "There are all kinds of simple problems that standard transformers cannot solve. An example is their ability to classify bit strings of arbitrary size according to whether the number of 1-bits is odd or even." Unfortunately, tossing more resources at the problem—scaling up models—will not fix the underlying problem, he noted.

k https://www.cs.cmu.edu/~aditirag/

Since 1990, Schmidhuber's research has centered on developing AI agents that learn a world model from data collected through "self-invented experiments" that identify regularities and patterns in the data. A controller uses simulations inside the world model to select action sequences that maximize long-term predicted rewards. "Goals change all the time because whatever the system learns becomes boring, and so it must continually invent new goals," he said. "This is essentially how babies and scientists learn—not by downloading the Web but through self-invented experiments."

LeCun is taking a different approach. His Joint Embedding Predictive Architecture (JEPA)[l] doesn't generate text or pixels directly; it uses images, video, and other signals to learn and predict abstract representations of future states through self-supervised learning. This JEPA-based world model resides inside an agent that has perception, memory, an actor, and a cost module. The agent uses JEPA's world model to simulate the consequences of actions and select those that appear most promising.

LeCun's goal is to escape the fundamental limitations of transformer models, including LLMs. "If you use a transformer model to predict future video frames, the results become a blurry mess because it has no idea what is actually taking place. As the system attempts to calculate an 'average,' it comes up with essentially nothing," he explained. Unfortunately, using a probability distribution model to solve the challenge becomes an "intractable mathematical problem" and plugging in a diffusion model "doesn't get us any closer to truly understanding reality."

JEPA uses sensor data from cameras, microphones, and an array of industrial or scientific instruments to identify the variables that actually matter and forecast how the scene will unfold. Its applications could range from robotics and aerospace to smart power grids and autonomous driving. In November 2025, LeCun announced he was leaving his position as chief AI scientist at Meta and

l https://arxiv.org/abs/2301.08243

Coming Next Month in COMMUNICATIONS

How Immigration and Federal Funding Sustain American AI Leadership

Agency as a New Organizing Abstraction for Computer Science

The 'Coloniality Problem' in AI and the Call for 'Decoloniality'

A Research Agenda for the Economics of Transformative AI

Toward a Theory of Data Ecology

Working on AI and Technology Policy in the U.S. Senate

Beyond Bits: Analog Computing for Real-Time Control

Web Scraping as Illegal Circumvention?

Tailoring Elective Courses to Enhance Women's Participation in Computing

Can VIBE Drive Development? Threats and Opportunities

Scalable and Personalized Oral Assessments Using Voice AI

launched a startup[m] to fully develop the JEPA framework. "The idea that we can solve complex AI problems purely from language is ridiculous. It will never work," he said.

What makes a world model so powerful is that it can play out a situation before generating a response. For example, in the coin and the mug scenario, an agent doesn't use language about these objects to produce a likely answer. It actually simulates the mug tipping, the coin leaving the mug, and the coin coming to rest on the counter before reporting where it ends up. Swap in a jar, add a second coin, or set the mug down twice along the way, and the simulation continues to track the objects. Likewise, a world model can project forward a few seconds and weigh the likelihood of a child following a ball into the street—along with the cost of a wrong decision.

In Search of Better Reasoning

Other post-transformer approaches are taking shape. For instance, researchers are exploring SSMs, such as Mamba,[n] which maintain a compact running "state" that is updated as new words or data arrive rather than constantly comparing every single word. They can process longer sequences—and at times use less compute power and memory—than current transformer models. This running state makes them an attractive option for physical prediction: a system tracking a flight trajectory or a weather front can carry forward past events rather than re-collecting and re-weighing every observation from scratch. SSMs could help AI better predict complex trajectories, weather patterns, and other dynamic events.

Google DeepMind is developing Titans,[o] a "neural memory" module that continues to learn and remember post-training. DeepMind researchers have also developed a technique called *Nested Learning*,[p] which reorients internal workflows to avoid catastrophic forgetting.[q] Meanwhile, AI21 Labs has developed Jamba,[r] a model that combines transformer elements with Mamba-style state spaces, while another group of researchers has created a framework called Retentive Network,[s] which takes aim at parallel training and recurrent low-cost inferencing.

"The idea that we can solve complex AI problems purely from language is ridiculous. It will never work."

For now, a post-transformer world is still taking shape—and it isn't clear when it will fully arrive or what form it will take. Some researchers, such as CMU's Raghunathan, expect the space to advance gradually. She believes that model designers will mix and match components to achieve desired performance benchmarks, build more flexible frameworks, and produce more grounded results. This will lead to "creative and open-ended problem solving" as well as "improved quality and reliability," she said.

Concluded Isola: "We can do a lot of things with transformer models—and there is likely a lot of room to make them perform better—but there are also inherent limitations in these models. We must find ways to build a more robust AI stack." C

m https://www.reuters.com/technology/meta-chief-ai-scientist-yann-lecun-plans-exit-launch-startup-ft-reports-2025-11-11/
n https://arxiv.org/pdf/2312.00752
o https://arxiv.org/abs/2501.00663
p https://research.google/blog/introducing-nested-learning-a-new-ml-paradigm-for-continual-learning/
q https://en.wikipedia.org/wiki/Catastrophic_interference
r https://www.ai21.com/jamba/
s https://www.microsoft.com/en-us/research/publication/retentive-network-a-successor-to-transformer-for-large-language-models/

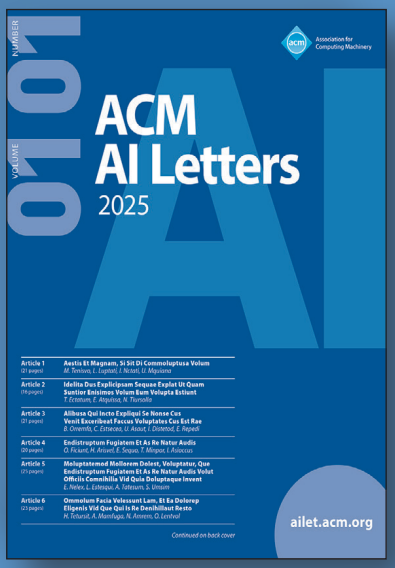
Samuel Greengard is an author and journalist based in West Linn, OR, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 ACM 0001-0782/26/10

NOW OPEN FOR SUBMISSIONS

ACM AI Letters (AILET)



Editors-in-Chief

Nitesh Chawla, *University of Notre Dame, USA*
Barry O'Sullivan, *University College Cork, Ireland*
Richa Singh, *IIT Jodhpur, India*

A new venue for rapid publication of impactful, concise, and timely communications in AI

ACM AI Letters (AILET) is a new venue for rapid publication of impactful, concise, and timely communications in AI. Bridging a crucial gap between traditional conferences and journals, AILET will feature short peer-reviewed contributions that accelerate knowledge dissemination across academia and industry. This unique publication prioritizes theoretical breakthroughs, algorithmic innovation, practical real-world applications, and critical societal implications, including ethics, policy, and responsible AI. It also introduces a distinctive space for rigorously reviewed opinion pieces and policy briefs, promoting swift engagement with contemporary issues shaping the AI landscape.

ACM AI Letters will welcome concise summaries of work in the following areas:

- Theoretical AI: Reports on theoretical breakthroughs in AI
- Algorithmic Advances: Descriptions of significant algorithmic and scientific advances in AI
- Practical Applications: Accounts of novel or deployed applications of AI in real-world settings such as healthcare, finance, robotics, and autonomous systems
- AI & Society: Reports on AI applications addressing key societal challenges, such as the United Nations Sustainable Development Goals
- Emerging Topics: Areas like AI ethics, explainable AI, policy, governance, and responsible AI will be covered, focusing on their implications in practical implementations
- Opinions and briefs: Policy, latest advances, comparative assessments

All *ACM AI Letters* papers will be published open access, with no publication charges for the first three years.

For more information, please visit ailet.acm.org

Association for Computing Machinery

DOI:10.1145/3785371

Josiah Hester

Opinion
Who Owns the Wai (Water)? AI, Power, and the Public Trust

From extractive infrastructure to AI in the public trust.

AI's impact is already hitting home, changing people's lives today. Across the U.S. and soon around the world, the rapid buildout of AI datacenters is driving up energy demand, draining water supplies, and reshaping communities—often without their consent. What was once framed as a technical scaling problem has become an environmental justice issue, playing out in zoning hearings, regulatory meetings, and town halls, as residents grapple with higher utility bills, constant noise, and contaminated or depleted water.

The shift into public view has been swift. Viral videos, front-page headlines, and local elections center on the real costs of AI infrastructure—rising electricity rates, water scarcity, air pollution, and falling property values. The promised benefits of AI—helping people live better, healthier, more productive lives—remain largely unrealized, even as its environmental and social costs are imposed on communities with little say in the matter.

Yet the industry's response is depressingly familiar: promises of future fixes, vague assurances of community benefit, and ever more fantastic claims about what AI will someday deliver for everyone. In practice, those promises are often paired with aggressive accounting and legal maneuvers that obscure present-day harm: land quietly purchased through shell companies to avoid public scrutiny, or state and local officials required to sign nondisclosure agreements before they are even allowed to ask basic questions about water use, power demand, or emissions. We have at least three reasons to question whether this infrastructure buildout is for the best.

What was once framed as a technical scaling problem has become an environmental justice issue.

First, for more than a decade, a small but persistent community has documented computing's—and especially AI's—escalating climate and environmental impacts, arguing that unchecked growth would lead to precisely this moment.[7] *Communications* itself echoed that warning in the Opinion column "Our House Is on Fire," rejecting "digital exceptionalism"—the belief that computing's benefits justify unlimited expansion.[5] Second, while AI is often promoted as a powerful tool for climate action—and it can be—its climate benefits must be weighed against both the direct costs of the infrastructure required to support it and the additional emissions enabled by AI-driven productivity gains, which recent modeling suggests may exceed the emissions AI helps avoid.[1] Investment and talent have largely shifted away from conservation and sustainability work toward scaling AI capabilities themselves. At the same time, the impacts that matter most to communities are

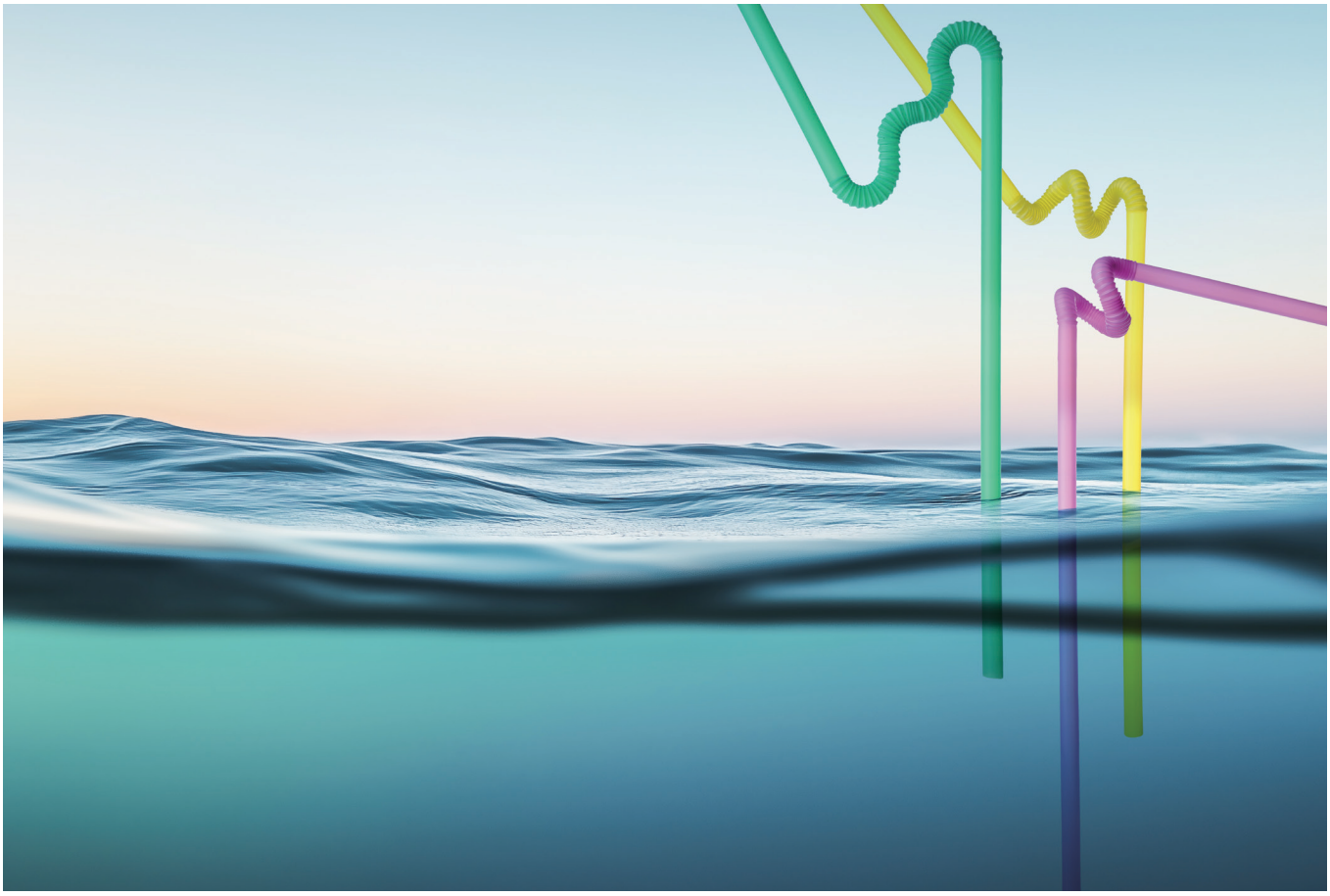


IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK.COM

systematically left out of industry reporting. Independent research from HuggingFace has shown that the metrics companies emphasize—energy use and greenhouse gas emissions—capture only a narrow slice of AI's footprint, ignoring water extraction, land use, air pollution, labor impacts, and community displacement.[6] When these broader harms are accounted for, the picture changes dramatically: Efficiency gains and AI-enabled climate tools are overshadowed by the physical and social costs of building and operating AI at scale.

Third, the question is not whether AI can be useful—it can—but whether we can build it without replicating familiar patterns of environmental injustice, where benefits flow upward to a privileged few while harms are imposed locally and endure. Given the incentives shaping today's tech industry, skepticism is warranted. As journalist Cory Doctorow has argued, digital systems tend to follow a predictable path of "enshittification": first optimized for growth, then for monetization, and ultimately for extraction at all costs.[3] That logic now governs AI infrastructure, where value is increasingly extracted not just from users as in the first Internet platforms (Facebook, Gmail, and so forth), but from the very land, water, air, and public resources that make up that user's world. There is little reason to believe these systems will become more accountable simply because companies promise they will.

The argument is this: The physical costs of AI need public governance. Who owns the infrastructure, who directs its growth, and who bears its costs cannot be left solely to private firms. Digital services are built from land, energy, labor, and water, or wai (the Hawaiian language word for freshwater, the most important resource on an island surrounded by ocean). Treating AI as private infrastructure with public consequences is no longer tenable if AI is meant to deliver shared prosperity. AI must be governed as part of the public trust: the principle that essential shared resources are held in stewardship for the public, protected from exclusive private capture, and managed for present and future generations. For AI infrastructure, that means transparency about water, energy, land, and emissions; requiring large users to bear the costs they impose on public infrastructure; meaningful community authority over siting and resource use; and public or community-controlled alternatives where critical infrastructure has become too concentrated. This idea is an ancient one. In Hawaii, Kānāwai—water law—recognized water as a communal trust to ensure all people could grow food, with dire consequences if someone took more than a fair share. Achieving this shift requires more than optimization. Computing professionals and institutions must build and defend new technical and structural alternatives—free and open source systems, repairable and maintainable hardware, distributed and community-controlled infrastructure, and mechanisms that align power with accountability—while standing alongside Indigenous and frontline communities as leaders in governance. I tell the story of wai through two stories from my home in Atlanta, GA, and my homeland of Hawaii, because they reveal what is al-

STUDENT RESEARCH COMPETITION

Association for Computing Machinery
Advancing Computing as a Science & Profession

ACM Student Research Competition

Attention:
Undergraduate *and* Graduate Computing Students

The ACM Student Research Competition (SRC) offers a unique forum for undergraduate and graduate students to present their original research before a panel of judges and attendees at well-known ACM-sponsored and co-sponsored conferences. The SRC is an internationally recognized venue enabling students to earn many tangible and intangible rewards from participating:

- **Awards:** cash prizes, medals, and ACM student memberships
- **Prestige:** Grand Finalists receive a monetary award and a Grand Finalist certificate that can be framed and displayed
- **Visibility:** meet with researchers in their field of interest and make important connections
- **Experience:** sharpen communication, visual, organizational, and presentation skills

Learn more:

https://src.acm.org

ready unfolding everywhere and offer us historical and current perspectives on the impacts of privatizing AI infrastructure. This moment of attention is an opportunity we must seize to determine AI, and our, future.

Your City: The Next Datacenter Megalopolis

During the past five years, the U.S. has experienced an unprecedented datacenter buildout, foreshadowing the global expansion now underway. In Atlanta and across the surrounding region, the transformation is reshaping the landscape. With approximately 1.46GW of datacenter capacity built, and another 2.08GW under construction, concerns about noise, property values, water availability, and power rates are on the minds of many. These developments carry real public health risks, including degraded air quality that disproportionately affects people with asthma.[11] In nearby Newton County, GA, national attention followed reports of household wells running dry and tap water turning brown after a new Meta datacenter came online (Meta disputes they caused well problems).

Communities have not been passive. Neighborhood planning units, local academics, and environmental groups have pressed city and state leaders to require datacenter developers to pay for the power infrastructure they need, rather than shifting those costs onto residents. Advocates have pushed for higher electricity rates for datacenters as large industrial consumers and have successfully blocked new facilities near dense neighborhoods and sensitive watersheds. Consumer groups such as the People's Power Union and Science4GA have challenged the outsized lobbying influence of Georgia's energy sector after years of rate hikes. Legislative victories, however, have been elusive: a bill to increase energy payments from new datacenters stalled despite broad support, while another to suspend generous tax breaks was vetoed by the governor.

The most consequential decision came in December 2025, when the Georgia Public Service Commission approved an unprecedented $15 billion, nearly 10,000MW expansion of power infrastructure to meet projected AI and datacenter demand. The vote occurred during the final days of a lame-duck commission—after voters had ousted two incumbent commissioners over affordability and energy concerns in a nationally covered election with record-breaking voter turnout, but before their replacements could be seated—leaving ratepayers potentially on the hook for decades of higher costs and increased climate burden. Since 2023, the commission has approved six Georgia Power rate increases, raising the average household bill by at least $43 a month, or roughly $500 a year, even as the company's profits rose by 40 percent. For many Georgians, the message is clear: the benefits of AI infrastructure flow elsewhere, while the costs are being paid at home.

Now the buildout is global. Singapore, despite its challenging climate and limited land, is converting land for datacenter use; in southern Italy, a "Mediterranean AI Hub" is being developed; and in India, Microsoft and others are investing billions in new datacenters. The scale of this expansion is staggering: the International Energy Agency (IEA) projects global datacenter electricity consumption to nearly double, from approximately 485TWh in 2025 to approximately 950TWh by 2030. Across these locations, consumer and environmental advocates are refusing to let Big Tech treat climate responsibility as a branding exercise or a series of accounting tricks. Efficiency improvements often trigger rebound effects: When a technology becomes more efficient and cheaper to use, total consumption tends to increase rather than decrease, ultimately exacerbating resource demand and environmental

The argument is this: The physical costs of AI need public governance.

impact.[9] In the context of AI and datacenters, making computing "greener" without limiting overall growth simply accelerates energy, water, and land use rather than reducing harm, even when offset by credits, which are often fraudulent.[10] Our own water, land, and tax dollars are used to support AI use everywhere except here. If allowed to continue, we as individuals, as communities, as parents, as employees, as renters, as homeowners, as citizens, will see our lives negatively impacted in the near term.

What Can We Do?

Data Center Watch, a research project from 10a Labs that tracks grassroots opposition to datacenter development across the U.S., reported that by early 2026 more than 800 active opposition groups were organizing across 49 states, with approximately $130 billion in datacenter projects blocked or delayed in the first quarter of 2026 alone through community meetings, classrooms, local policymaking, and activism led by those who often bear the brunt of climate impacts: Indigenous communities, Black communities, and other communities of color.[a] It is clear that communities are pushing back, and we should be paying attention.

Our key question is how we, as computing professionals, support this grassroots effort. I take hope from people already using computing in radically different ways—practitioners working alongside environmental justice groups, leveraging technology for genuine public good. This is not a new fight: Decades ago, communities pushed back against landfills being placed only in Black neighborhoods, and now we are seeing similar battles play out over the placement of datacenters and the draining of local resources. Meaningful change requires collective action that moves beyond technical optimization and brings together consumers, policymakers, and community organizations to force industry shifts. What is needed now are morally ambitious people willing to steer the direction of computing and AI at every level. So what can be done? Several existing efforts point toward what this can look like:

What is needed now are morally ambitious people willing to steer the direction of computing and AI at every level.

▸ **Community-governed, place-based computing for stewardship**. Direct partnerships such as the STRONG Manoomin Collective in the Great Lakes region and the CEAR Hub in Brunswick, GA, show what community-governed, place-based AI can look like in practice. Working alongside federally recognized Ojibwe tribes, STRONG Manoomin develops sensing and AI systems to preserve wetlands and assert tribal sovereignty, utilizing data for stewardship, advocacy, and cultural restoration, rather than as benchmarks.[8] In Brunswick, youth and researchers co-deploy water-level sensors, climate curriculum, and flood modeling in neighborhoods experiencing rising flood risk. In both cases, computing is embedded in local knowledge and lived experience, serving community resilience rather than extractive innovation.

▸ **Indigenous-led technology and ancestral stewardship.** In Hawaii, the Purple Maiʻa Foundation works at the intersection of Indigenous knowledge, computing, and entrepreneurship, co-developing technologies with Native Hawaiian youth and communities to restore and protect ecosystems. Their programs train a new generation of technologists grounded in ancestral stewardship. The Nation of Hawaii, too, leverages sensors, mapping tools, and networked infrastructure to monitor and protect ʻāina (land) from overdevelopment. At a recent event, called Build4ʻĀina, computing professionals from Big Tech companies, academic computing researchers, and local artists and practitioners gathered to build AI and sensor systems to directly support community conservation, traditional food production, and community action, eventually demoing these systems to state government agencies.

▸ **Carbon transparency and literacy for developers:** Tools such as GreenPT and CodeCarbon can help developers see the resource costs of AI workloads, challenging the illusion that computing is zero-impact.

These efforts need more hands—now. Every computing professional has a role to play in shifting AI from extractive infrastructure to community-governed systems, tools for resilience, land stewardship, and self-determination, not harm.

Lessons from Hawaii's Past for AI's Future

I conclude with a warning from the past, and a vision of hope for our shared future. My people (Native Hawaiians) have a proverb taught to almost every child: *I Ka Wā Ma Mua, I Ka Wā Ma Hope*—the future is in the past. Other indigenous cultures describe this as moving backward into the future—the belief that past experiences, traditions, ancestral knowledge, and learning will guide us today in making forward-looking and just decisions.

My Polynesian ancestors voyaged across the Pacific to the Hawaiian islands more than 1,000 years ago and built extraordinary, whole-island wai engineering works called ahupuaʻa to sustain themselves. They developed a system of laws called Kānāwai, which ensured a careful, place-based framework for food production, water protection, and people's well-being. The fresh water stream was life, and wealth—called waiwai, as in much water—was measured not in extraction or accumulation, but in the ability of water to support everyone. At its peak, this system supported more than 700,000 people's needs; today, most food is imported to Hawaii, and the Hawaiian islands and their people are at extraordinary risk from climate degradation.

The story of Hawaii's transformation from a self-sufficient, 1,000-year-old advanced sovereign nation to a subjugated state with the highest cost of living in the U.S., and dire food and water insecurity, is a complex one.

a See https://www.datacenterwatch.org/q1-2026.

However, one particular period in the Hawaiian Kingdom's history mirrors our current time.

Water as a public trust resource was codified in written law in 1840. Even as sugar plantations expanded and profits from foreign investors overrode local concerns, systems were in place to regulate water withdrawals (via large ditches and built infrastructure) to ensure that small farmers could access water alongside the enormous sugar plantations. Water commissioners from the region were tasked with weighing impacts, scheduling withdrawal times, and preventing total enclosure of resources. The system was rooted in Hawaiian values that emphasized the perpetuation of land and water for future generations, as well as the local knowledge that the water cycle influenced everything from wildfire frequency to fisheries and lei making. Disruption would harm all; water was a public trust, not a commodity. My own ancestors were part of the commissioning body that adjudicated water management at the time, listening to oral testimony from farmers, plantation representatives, and longtime residents, to ensure fair use. Often, the stream diversions benefited small farmers, who could take advantage of the large infrastructure projects undertaken by sugar plantations. The Hawaiian kingdom leveraged revenue from sugar plantations during this time to build public works, educate the population, pay for roads, schools, and hospitals, ensuring the public trust.[2]

This robust public trust framework collapsed half a century later with the overthrow of the monarchy by sugar plantation leaders supported by the U.S. Marines, eager to exploit a fraught time in global history, and ride industrialization to drive huge profits. The consolidation of rights over nearly all water on the islands into the hands of "The Big Five," descendants of missionary families and other businessmen, concentrated in corporations, wrested control away from community stewardship. Sugar production skyrocketed, and water became scarce (it takes approximately 1,000 liters of water to produce one pound of sugar). By the time plantations eventually shut down, well into the 20th century, the water and land were already broken. Today, Hawaii imports most of its food, and the legacy of diverted streams still haunts the islands.

I see in this history a warning, a guide for how we govern AI.

I see in this history a warning, a guide for how we govern AI. Hawaii's Big Five consolidated the infrastructure of sugar—the land, water, finance, and transportation on which the industry depended—and with it, the power to redirect shared resources toward private profit. Today's hyperscalers are consolidating the infrastructure through which AI is built and delivered. At the end of 2025, hyperscale operators controlled nearly half of global datacenter capacity, and Amazon, Microsoft, and Google alone accounted for roughly three-fifths of hyperscale capacity. The danger is not concentration alone, but what control of infrastructure allows its owners to control upstream: access to energy, water, land, public investment, and ultimately the terms under which communities participate in the AI economy. As that control concentrates, decisions about shared resources move away from the communities that bear their consequences. Just as plantations diverted streams to maximize production while leaving communities dry, today's tech giants divert energy, water, and public resources to feed their own system. If we build AI the way sugar captured water—extractive, consolidated, and dismissive of community voice—we will inherit the same legacies of scarcity, inequity, and disempowerment. But if we treat AI like wai, belonging to the public trust of land, water, and air, we can chart a different course.

British-American computer scientist and technology entrepreneur Andrew Ng said, "AI is the new electricity." I disagree. AI is water, AI is wai. Literally, because we transform water into computation in datacenters, and figuratively, because AI, like water, should be a public trust resource, available to all. In Hawaiian, waiwai means both water and true wealth. AI can become waiwai—shared abundance. To avoid repeating Hawaii's history, we as computing professionals must empower and collaborate with communities, especially Indigenous ones, who have safeguarded water and other vital resources for generations, reimagining these systems, harnessing this extraordinary moment of consciousness about AI's impacts, to build a better future for us all. C

References

1. Alpine, W. et al. AI-driven productivity gains enable more CO_2 emissions than they avoid in a global energy–economy model. *NPJ Clim. Action 5*, 71 (2026); https://doi.org/10.1038/s44168-026-00411-0.
2. Beamer, K. *Waiwai: Water and the Future of Hawai'i*. University of Hawaii Press (2025).
3. Doctorow, C. *Enshittification: Why Everything Suddenly Got Worse and What to Do about It*. Verso Books (2015).
4. Gomes, C. et al. Computational sustainability: Computing for a better world and a sustainable future. *Commun. ACM 62*, 9 (Sept. 2019), 56–65.
5. Knowles, B. et al. Our house is on fire: The climate emergency and computing's responsibility. *Commun. ACM 65*, 6 (May 2022), 38–40; 10.1145/3503916.
6. Luccioni, A.S., Strubell, E., and Crawford, K. From efficiency gains to rebound effects: The problem of jevons' paradox in AI's polarized environmental debate. In *Proceedings of the 2025 ACM Conf. on Fairness, Accountability, and Transparency* (2025), 76–88.
7. Nardi, B. et al. Computing within limits. *Commun. ACM 61*, 10 (Oct. 2018), 86–93.
8. Rothrock, B. et al. Makak: Co-designing environmental sensors to protect manoomin (wild rice). In *Proceedings of the 2025 ACM SIGCAS/SIGCHI Conf. on Computing and Sustainable Societies (COMPASS '25)* (2025), 115–130; 10.1145/3715335.3735460.
9. Sorrell, S. et al. The Rebound Effect: An assessment of the evidence for economy-wide energy savings from improved energy efficiency. *UK Energy Research Centre* (Oct. 2007).
10. West, T.A.P. et al. Action needed to make carbon offsets from forest conservation work for climate change mitigation. *Science 381*, 6660 (2023), 873–877; 10.1126/science.ade3535.
11. Wierman, A. and Ren, S. We need to talk about AI's impact on public health. *IEEE Spectrum* (2025); https://spectrum.ieee.org/data-centers-pollution.

Josiah Hester (josiah@gatech.edu) is an associate professor of Interactive Computing and Computer Science, the director of the Center for Advancing Responsible Computing, and the associate director for Civic Innovation and AI in the Brook Byers Institute for Sustainable Systems (BBISS) at the Georgia Institute of Technology, Atlanta, GA, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

Watch the author discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/who-owns-the-wai



DOI:10.1145/3838602

Michael A. Cusumano

Technology Strategy and Management

The ASML Monopoly: Platform and Ecosystem Leadership in EUV Lithography

Scratching the surface of the leading manufacturer of extreme ultraviolet lithography technology used for manufacturing integrated circuits.

GENERATIVE AI SOFTWARE requires a semiconductor stack that is increasingly constrained at multiple levels: Nvidia dominates GPU designs.[8] TSMC dominates advanced chip manufacturing.[9] High-bandwidth memory is dominated by SK Hynix, Micron, and Samsung. But another critical bottleneck is one layer deeper: the equipment used to fabricate the most advanced chips. At the center of that other layer is a company known as ASML.

Based in the Netherlands, ASML (the acronym name is derived from Advanced Semiconductor Materials Lithography) is Europe's most valuable technology company. It produces approximately 90% of all lithography tools and is the *only* supplier of extreme ultraviolet (EUV) systems, a critical manufacturing technology.[29] ASML has a market value of approximately $677 billion, with more than $37 billion in FY 2025 revenues, $11 billion in profits, and 44,000 employees. But the company's de facto monopoly is not based on a single breakthrough innovation. Rather, it results from decades of foresighted investment combined with extraordinary platform leadership and ecosystem management skills.[11] No competitor has yet been able to reproduce the core technology or the full stack of system capabilities needed for EUV lithography.



IMAGE © ASML. COURTESY OF ASML MEDIA LIBRARY

Since 2019, following U.S. policy, the Netherlands has prevented export to China of the most advanced ASML machines. This ban has provoked in China a Manhattan Project-type of response. This column examines how ASML has come to dominate EUV lithography, why the company and the technology have been so difficult to replicate, and what China is doing about it.

Technology and Company Origins

Photolithography uses light to "print" billions of microscopic circuit patterns onto silicon wafers treated with photosensitive coatings. The technology resembles photography and dates

ACM Transactions on Accessible Computing

ACM TACCESS is a quarterly journal that publishes refereed articles addressing issues of computing as it impacts the lives of people with disabilities. The journal will be of particular interest to SIGACCESS members and delegates to its affiliated conference (i.e., ASSETS), as well as other international accessibility conferences.



For further information or to submit your manuscript, visit taccess.acm.org

back to the 1950s, invented for the U.S. military and commercialized by firms such as Fairchild Semiconductor and Texas Instruments. By the early 1980s, Geophysical Corporation of America (GCA) had as much as an 85% market share in lithography machines. By the later 1980s, however, Nikon and Canon overtook GCA (which shut down in 1993) and came to dominate the industry.[19,38]

ASML began in 1984 as a joint venture between Philips, the Dutch electronics conglomerate founded in 1891 (and an early investor in TSMC), and ASM International, a small Dutch semiconductor manufacturing equipment firm founded in 1964. The goal was to commercialize Philips' in-house photolithography technology. However, ASML's first product, the PAS 2000 (PAS stands for "Philips Automatic Stepper"), was unreliable and a failure.[35] In 1988, ASML became independent. Its next product line, the PAS 5500, introduced in 1991, was highly successful, using a modularized architecture, precision optics from Carl Zeiss AG of Germany (based on a partnership started in 1983), and "near" ultraviolet light to produce denser chip designs.[2,39] PAS 5500 in the mid-1990s added "Deep Ultraviolet" (DUV) capabilities, which the Japanese were also perfecting.

The challenge for European and U.S. lithography producers in the 1990s was to find a way to leapfrog Nikon and Cannon. One way was to establish an early leadership position in the next generation of the technology.[12] DUV had physical limits and experts believed the shorter wavelengths and higher photon energy of EUV would be necessary to produce denser chips. The problem was how to generate and control EUV light for mass production. There were other candidate technologies (X-rays, electron beams, ion beams, and charged particles), but none offered the potential speed and cost advantages of EUV.[17]

Successful commercialization of EUV lithography is one of the greatest stories in the history of science, engineering, and business. That said, the success took roughly two decades, $11 billion in R&D, and a global effort coordinated by one firm—ASML.[24] The history goes back to 1985, when Hiroo Kinoshita, a researcher at Japan's NTT labs, adopted multilayer mirror research from the Soviet Union to project the first EUV images. He used Mercury-vapor lamps, which produced ultraviolet light of 365nm. Kinoshita's presentation at a 1986 Japanese optics conference was met with skepticism but prompted other researchers around the world to study EUV lithography.[13]

By 1994–1995, ASML executives had concluded EUV was indeed the future of lithography. Martin van den Brink (later president and CTO), a technology executive with a strong physics background, had led development of the PAS 5500. He championed the effort to explore EUV and put another senior manager, Jos Benschop, in charge.[13,34] In 1998, ASML and Zeiss formed a European industrial consortium named EUCLIDES for "Extreme UV Concept Lithography Development System."[13] Similar consortia appeared in the U.S. and Japan, but only EUCLIDES succeeded.

In 2000, convinced they could overcome the scientific hurdles, ASML created a small team to build a prototype EUV stepper. ASML also benefitted from U.S. military R&D after it bought the last surviving U.S.-based lithography company, Silicon Valley Group (SVG), also in 2000.[19] The prototype took six years, completed in 2006. ASML worked closely with key suppliers and universities as well as IMEC, the Interuniversity Microelectronics Centre in Belgium, which served as a laboratory and equipment testing ground for the European semiconductor industry. In 2009, ASML opened a new EUV R&D facility at its headquarters in Veldhoven, Netherlands.[13]

Another important milestone occurred in 2012. Three key customers—Intel, along with Samsung and TSMC—agreed to become investors in ASML to share the R&D costs and help resolve the technical challenges.[13] Intel's investment of $4 billion (a 15% stake—several times larger than TSMC and Samsung) was especially significant.[18] Intel was the world's largest semiconductor manufacturer, and this decision signaled a major commitment to EUV. ASML began shipping EUV prototypes in 2010 but needed help to produce a commercially

viable system, which it did by 2013. Machines perfected for mass production took until 2017 and were priced at $100 million.[3,29]

Despite its early backing of EUV, Intel decided to delay commercial introduction until the next generation because of the high cost. Unfortunately, this delay (until 2025) caused Intel to fall behind TSMC and Samsung in advanced manufacturing capabilities. It also has led to severe financial and reputational difficulties, from which Intel is only now starting to recover.[1,7] Samsung would go on to ship the first EUV-enabled product (using what is referred to as a "7nm" node process) in August 2019 for its Galaxy Note 10 smartphone.[13] TSMC shipped its first EUV-enabled products (also with the 7nm node process) in October 2019 for multiple customers, including Huawei in China.[30]

Why ASML's EUV Machines Have Been So Difficult to Copy

In semiconductor manufacturing, to shrink circuit feature sizes depends fundamentally on the wavelength of light used in lithography: Shorter wavelengths enable finer patterns and denser packing of transistors. For decades, the industry has relied on DUV systems operating at 193nm, and these machines are still widely used. EUV reduced the potential wavelength to 13.5nm. This is a dramatic improvement, though mass production required significant advances in a range of complementary technologies.[14]

For example, EUV light cannot pass through air or conventional glass, which eliminates traditional lens-based approaches. Instead, ASML's EUV machines rely on highly specialized multilayer mirrors, fabricated with sub-nanometer precision by Zeiss, to reflect and focus the EUV light. Generating and controlling EUV light is another complex physics problem. ASML uses a high-powered laser produced by TRUMPF GmbH, a German company, to strike microscopic tin droplets at 50,000 times per second. This creates a plasma about 200,000 degrees centigrade (40 times hotter than the surface of the sun) that emits the required wavelength.[23] The laser module alone is said to contain 457,329 parts.[19]

The wafer masks are another critical component. Semiconductor foundries create the masks but work closely with electronic design automation (EDA) tool companies such as Cadence and Synopsys. The chip manufacturers use EDA tools to generate data for the 50 or so alternating layers of silicon and molybdenum in each mask, which must be perfectly compatible with the EUV optics, light system, and the stepper's numerical aperture.[10] To make sure each EUV machine can print the mask patterns and produce functioning semiconductor devices also requires special test engines and many rounds of simulations, done by the EDA firms and the foundries.[33] Other firms handle final assembly and packaging of the semiconductor devices, with processing steps almost as complicated as the core EUV technology.[6]

The challenge for potential competitors is that no one component defines performance of the whole system.

The challenge for potential competitors is that no one component defines performance of the whole system. The light source, optics, wafer, mask, and control software all come from different specialist firms, and their modules must all function as a tightly coupled system and operate in a vacuum under extreme thermal, mechanical, and computational constraints. Each component has required scientific and engineering advances in different fields, achieved over decades, in plasma physics, precision optics, mechanical engineering, materials science, and software engineering. ASML also holds the rights to thousands of patents that protect its technology.[5]

EUV machines weigh approximately 400,000 pounds, are the size of a double-decker bus, and require three Boeing 747s to transport.[20] ASML charges around $200 million for less sophisticated versions and $400 million for the most advanced versions ("2nm" and denser node processes). Each system contains more than 100,000 modules from some 800 suppliers, which have thousands of additional suppliers in their supply chains. ASML's strategy has been to outsource 90% of the modules and act as the R&D leader and system integrator.[20,28] This strategy leveraged the R&D and capabilities of many outside firms, but has limited ASML's control over manufacturing capacity. In 2025, ASML shipped just 48 EUV systems out of 327 ASML lithography machines of all types.[16] The company plans to ship 60 EUV systems in 2026 and 80 annually in future years.[15]

EUV machines and advances in chip design have enabled the semiconductor industry to continue on the path laid out by Moore's Law and produce increasingly advanced microprocessors, GPUs, and memory chips. The price tag limits the market for EUV machines but, as Intel discovered, due to their power, precision, and speed, using EUV to make the densest chips has become cheaper than workarounds with creative three-dimensional chip designs and DUV equipment.[34]

The cost and complexity of ASML's EUV machines help explain why the industry's leading manufacturing firms—Intel, TSMC, and Samsung—chose not to develop EUV lithography themselves but instead to invest in ASML. The cost and complexity also help explain why companies that mastered DUV lithography—particularly Nikon and Canon—gave up on EUV. They also fell behind ASML in advanced DUV machines, where ASML holds between a 60% and 85% market share.[20,31] It probably had some impact that the Japanese did not have direct access to U.S. military and national laboratory research, which ASML did through its SVG acquisition.[19]

Future Competition: Spotlight on China

ASML illustrates a unique type of monopoly—a dominance rooted not in a single proprietary invention but in a technology platform and ecosystem strategy used to integrate a complex system of components and skills provided by hundreds of partners, suppliers, and customers. ASML subsidiaries

also maintain tight control over critical technologies: HMI—mask pattern verification, Cymer—light sources, and Berliner Glas—optics.[25] This platform-ecosystem structure is useful to understand why ASML's main customers became investors and why competitors face a challenge that looks increasingly like a nation-state effort. The Russians have been trying to develop EUV technology for years but appear to be as much as two decades behind ASML.[4] The Chinese are more advanced and investing heavily, spending an estimated $50 billion per year on semiconductor manufacturing technology.[25]

China's leading foundry, Semiconductor Manufacturing International Corporation (SMIC), founded in 2000 as an attempt to build a TSMC-like foundry in China, had been a large ASML customer.[19] However, SMIC and other Chinese firms since 2019 have been prohibited from importing not only EUV machines but also the latest DUV machines produced by ASML as well as Canon and Nikon. SMIC and other foundries have extended DUV technology by using "multi-patterning" to add extra lines on the masks with multiple passes. Though costly and slow, this approach has enabled the production of 7nm and 5nm node devices, which SMIC has done mainly for Huawei.[10,27]

To produce its own EUV machine, China launched a project that is supposed to be a state secret. As reported by Reuters, in early 2025, former Chinese ASML engineers completed a prototype at a Shenzhen laboratory using components from older ASML machines available in secondary markets. The prototype is operational and undergoing testing. Observers believe it will take until at least 2030 for commercial production.[22,26,37]

China's effort differs from simple reverse engineering. The Central Science and Technology Commission is overseeing the project, with Huawei as the system integrator. The Shanghai Institute of Optics and Fine Mechanics (SIOM) as well as several other private companies are involved as well. They are using solid-state lasers for EUV light generation, an approach ASML determined was inadequate for mass production. The Chinese still need Zeiss mirrors (also banned for export to China) or a substitute. In addition, Huawei is developing a Laser-Induced Discharge Plasma (LDP) technology that uses high voltages to generate extreme ultraviolet light.[32] LDP should use less energy than ASML's tin-based approach but still generate the 13.5nm EUV light.[36]

Perhaps the biggest challenge for China is to produce EUV machines that are suitable for mass production.

Perhaps the biggest challenge for China is to produce EUV machines that are suitable for mass production, though initial cost is probably a secondary concern. For example, for 7nm chips using DUV technology, SMIC reportedly had yields less than 50% and wafer costs 40% to 50% more than TSMC, but at least it was able to make these advanced devices.[32] Another challenge is that the target is moving: ASML and its customers are already introducing technology for sub-2nm fabrication.[21,29] At some point, the laws of physics will limit ASML's approach and a better technology will emerge, but we are not there yet. C

References

1. ASML. *Intel and ASML Strengthen Their Collaboration to Drive High-NA into Manufacturing in 2025*. Company Press Release (Jan. 19, 2022).
2. ASML. *The Origin Archive: Breakthrough* (Chapter 3) https://theoriginarchive.com/company/asml/chapter/3.
3. ASML. *The Origin Archive: Transformation* (Chapter 4) https://theoriginarchive.com/company/asml/chapter/4.
4. Astute. Russia details domestic EUV lithography roadmap to 2037 in bid for chipmaking self-reliance. *AstutegroupCom* (Oct. 3, 2025).
5. BTFS. ASML competitors in 2025: Who's really challenging the semiconductor giant? *Backtofrontshow.com* (June 19, 2025).
6. Clark, D. How a niche technology became a choke point for A.I. *New York Times* (June 26, 2026).
7. Cusumano, M.A. Intel's fall from grace. *Commun. ACM 68*, 1 (Jan. 2025), 30–32.
8. Cusumano, M.A. Nvidia at the center of the GenAI ecosystem—for now. *Commun. ACM. 67*, 1 (Jan. 2024), 33–35.
9. Cusumano, M.A. TSMC: Manufacturing as an innovation platform. *Commun. ACM 69*, 7 (July 2026), 40–42.
10. Eureka-Patsnap. Inside an EUV mask: Multilayer structure, absorber, and pellicle explained, *Eureka.patsnap.com* (July 28, 2025).
11. Gawer, A. and Cusumano, M.A. *Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation*. Harvard Business School Press, Boston (2002).
12. Hijink, M. *FOCUS: The ASML Way*. Uitgeverij Balans, Netherlands (2023).
13. Hofman, S. Making EUV: From lab to fab. (Mar. 30, 2022); https://www.asml.com/en/company/stories/2022/making-euv-lab-to-fab.
14. Hwang, R. ASML: The little-known source of the world's technological progress. *Michigan J. Economics* (Apr. 2023).
15. Jacobs, S. ASML plans to build at least 60 EUV machines this year as AI chip demand surges. *Techspot.com* (Apr. 27, 2026).
16. Jaimes, R. ASML statistics 2026. *Quantumun.com* (May 9, 2026).
17. Levinson, H.J. *Principles of Lithography*. SPIE Digital Library (2011).
18. McGrath, D. Intel buys stake in ASML to boost 450-mm EUV R&D. *EE Times* (July 8, 2012).
19. Miller, C. *Chip War*. Scribner, New York (2022).
20. MindRemix. The ASML monopoly: How one Dutch company controls every advanced computer chip. *mindremix.com* (Feb. 6, 2026).
21. PatSnap Insights. ASML EUV and High-NA lithography technology roadmap, *Patsnap.com* (Apr. 2, 2026).
22. Potkin, F. How China built its 'Manhattan Project' to rival the West in AI chips. *Reuters.com* (Dec. 18, 2025).
23. Putzu's World. ASML EUV lithography: Interactive explainer. *Putzu.com* (2026).
24. Semiconductorx. ASML supply chain spotlight. *Semiconductorx.com* (2026).
25. Shubham, K. ASML competitors: Why Intel and Samsung can't catch up. *Growthnavigate.com.* (June 15, 2025).).
26. Stonor, J. ASML shares slip on report of U.S. concern that China has its tech. *Wall Street J.* (June 19, 2026).
27. Tech Insights. SMIC N+3 confirmed: Kirin 9030 analysis reveals how close SMIC is to 5nm. *Techinsights.com* (Dec. 11, 2025).
28. The Decision Magazine. Europe's Answer to Silicon Valley: The Rise of ASML, *thedecisionmagazine.com* (Jan. 11, 2024).
29. Thompson, C. The $400 million machine powering the future of chipmaking, *MIT Technology Rev.* (June 23, 2026).
30. TSMC. *TSMC's N7+ Technology is First EUV Process Delivering Customer Products to Market in High Volume*. Company Press Release (Oct. 7, 2019).
31. TwinTowerCity. ASML vs. Nikon vs. Canon—Semiconductor lithography equipment. *Machinehours.com* (June 1, 2026).
32. Valerio, P. China EUV breakthrough and the rise of the 'silicon curtain.' *EE Times* (Dec. 23, 2025).
33. Van Bocxlaer, A. EUV lithography in semiconductor manufacturing. *Think WIOT* (Feb. 1, 2022).
34. van den Brink, M. *Oral History*. Computer History Museum (Sept. 16, 2025); https://www.youtube.com/watch?v=XK84TlYTv5k.
35. Van Duijn, J. ASML's founding story: Our roots in the semiconductor industry. (Apr. 13, 2024).
36. Van Klinken, E. ASML expands in China, but Huawei emerges as EUV challenger. *Techzine.eu/blogs* (Mar. 10, 2025).
37. Van Klinken, E. How close is China's EUV project to eliminating ASML's monopoly? *Techzine.eu/blogs* (Dec. 18, 2025).
38. Jon, Y. How Japan won the lithography industry. *The Asianometry Newsletter* (May 24, 2023).
39. Zeiss Group Global. *History of Lithography Optics: How It All Began* (2026); https://www.zeiss.com/corporate/en/about-zeiss/past/history/technological-milestones/lithography-optics.html.

Michael A. Cusumano (cusumano@mit.edu) is the SMR Distinguished Professor and former Deputy Dean at the Massachusetts Institute of Technology Sloan School of Management, Cambridge, MA, USA, and coauthor of *The Business of Platforms* (2019).

For their comments, he thanks David Yoffie and Duane Boning.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© ACM 0001-0782/26/10



DOI:10.1145/3837871

George V. Neville-Neil

Development of this article led by acmqueue
queue.acm.org

Kode Vicious
Pebbles on the Beach

Keeping hold of the big questions.

Dear KV,

I know that your column is mostly for practitioners, but I have a question about academia for you. So, I hope it is OK if I throw you a bit of a curveball question. I have been working as a research engineer in a university computer science department. This is my first job out of college—although "out" is a bit wrong since I am still at the same institution where I got my undergrad degree. I thought I would spend some more time here building things for people while I decide whether to pursue a graduate degree or go into industry. Everyone seems very worried about LLMs and AI and all that, but I did a senior project with a professor here and she had funding for a position for a year and asked if I wanted to stick around. So, I did. The work is a continuation of the project I was already on, only now I am better paid and share an office with a grad student who is also working on the same project.

One of the things I have been trying to pick up, along with some new data analysis and scientific Python skills, is how researchers go about choosing or finding problems to study. There's a group lunch each week, and the grad students present what they are working on and get comments from my boss, along with a few other professors. But it all seems very open ended, as if they are just making it up as they go along. I even asked the grad student I am sharing the office with, and his response was not quite as obnoxiously Zen as something from Yoda,



IMAGE BY MARIA MENSHOVA

but it was something like, "I just wander around the space picking up rocks and looking under them until I find a problem that interests me, and then I build some small prototype to see if the idea has merit." When I pressed him on how he judged merit, the best I got was, "Can I think of a journal or conference that would publish what I want to write about?"

Now I have this image of researchers wandering around with colorful sand pails, picking up pebbles on a beach, and seeing which ones they want to take home. Since I do not have any other academic experience to compare it with, I wonder if this pertains to just this group or is more widespread. I know from your bio that you work with universities. So, is this common, or is it just my group?

Pebbles on the Beach

Dear Pebbles

You read my bio? What did it say? Hopefully nothing too salacious, although cavorting with academics might be considered ... well, never mind.

On the one hand, I could be equally

vague and say, "There are as many intellectual styles among academics as there are grains of sand." But, in fact, that has not been KV's experience in his time as someone who has done, as you do, development for research.

Research is often billed as being "curiosity driven." And, with the recent attacks on academia, particularly in the U.S., this phrasing has even made its way into the popular press. For the whole of human history, some people have been curious about the world around them, and they have come up with different ways to satisfy that curiosity. Newton is said to have stuck a needle around his eyeball to "learn about colour" (his spelling, not mine). You have to really be interested in how your eye works to mess with it in that way.

Now, the real question is: Was Newton just idly walking one day when he thought, "Cheerio, if I shove a needle into my eye socket, will I get a minimum publishable unit for the *Journal of Crazy Eyeball Physics and Optics*?" I kind of hope this was not what he was thinking, and I also kind of hope Newton didn't sound like some parody of Monty Python either. Rather, KV hopes Newton had a question he was trying to answer about the nature of light, or the nature of how we perceive light.

This is where we come to what KV has seen from the best researchers: They are not just wandering around picking up shiny pebbles; instead they are trying to answer some particular question. Answering questions is what scientists—and engineers—are supposed to do. Which provides us with a crossover between science and practice.

When KV is debugging a piece of code (which is most of what he does, since he works on large codebases written by 12 monkeys with ASR-33 terminals who were trying to hammer out a copy of Version 7 Unix by typing randomly), he too is trying to answer a question: Why does this function return garbage but only after more than a million correct operations?

We see here the effect of the question on the eventual solution. This is why the polestar KV uses to keep him on track is: What is the problem we are trying to solve?

Answering questions is what scientists—and engineers—are supposed to do.

Should you ever lose track of this guiding star, you will get lost in a universe of dull red dwarfs, bright new blue suns, neutron stars, and the most dreaded star of all—the black hole.

It has been my privilege over the years to work with some academics for whom this question of focus is always firmly in mind. They may have ideas about how to answer a question, in part, because of happenstance—the equivalent of picking up a shiny pebble on the beach is very much the "Aha!" moment in the shower, but this is not what is guiding them. The guiding star is a question such as: Why do all systems seem to reduce to this one type of system? What is the optimal layout of this set of data structures? How can we build an interface that is usable by nonspecialists and yet does not impede the specialist user? And, of course, since KV is a "systemsy" person, How can I make this system perform more operations per unit of time than it already performs?

If KV had been a physicist, which might have been the case had he been smarter, then the questions would have been grander: Where did the universe come from? Can anything travel faster than light? And the perennial, Where is my other sock? I put a pair in the dryer and now I only find one.

There are many advantages to this question-focused form of research, not the least of which is that it can lead to a longer-term program than a single, shiny pebble might. KV was very young when he was first exposed to the people building BSD Unix at UC Berkeley, a project they had already been working on for many years, which had not only produced useful working systems for others, but also led to answers to several questions about software design, process scheduling, and file storage, among others. While the group might not have used the same terms KV is using here, one can see that the questions they asked led to much of the research they produced. KV continues to seek out researchers who are question-focused in this way because questions also lead to narratives, and KV hates work without a narrative arc.

The need for a narrative arc is a personal bias. Not everyone needs one, but there are fewer things more maddening than watching someone wander around a space bumping into ideas as if they are in a pinball machine, learning nothing more than a few words that might be used at a cocktail party to impress the guests. To develop ideas that coherently answer questions, there must be order—and that order is the narrative arc.

If you find that you as well need this form of order to organize thoughts and you like to keep a good question in front of you to guide you, then, by all means, continue your work with academics. Look for the ones who don't just publish papers on a particular topic, but those who develop full programs that attract bright graduate students who understand the *research question*, as they call it, and then take a piece of that question and try to answer it. This is the way in which the edifice of science has been built. Picking up pebbles on the beach is a hobby, but answering questions is a calling.

KV

Related articles

You Don't Know Jack About AI
Sonja Johnson-Yu and Sanket Shah
https://queue.acm.org/detail.cfm?id=3703128

The Eternal Promise
Ivan Turkovic
https://www.ivanturkovic.com/2026/01/22/history-software-simplification-cobol-ai-hype/

Does AI Now Represent a Paradigm Shift?
Vinton G. Cerf
https://cacm.acm.org/opinion/does-ai-now-represent-a-paradigm-shift/

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

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3788880

Christoph Mueller-Bloch and Raffaele Fabio Ciriello

Opinion
Governing Social Media as a Public Utility

A case for sovereign digital infrastructure.

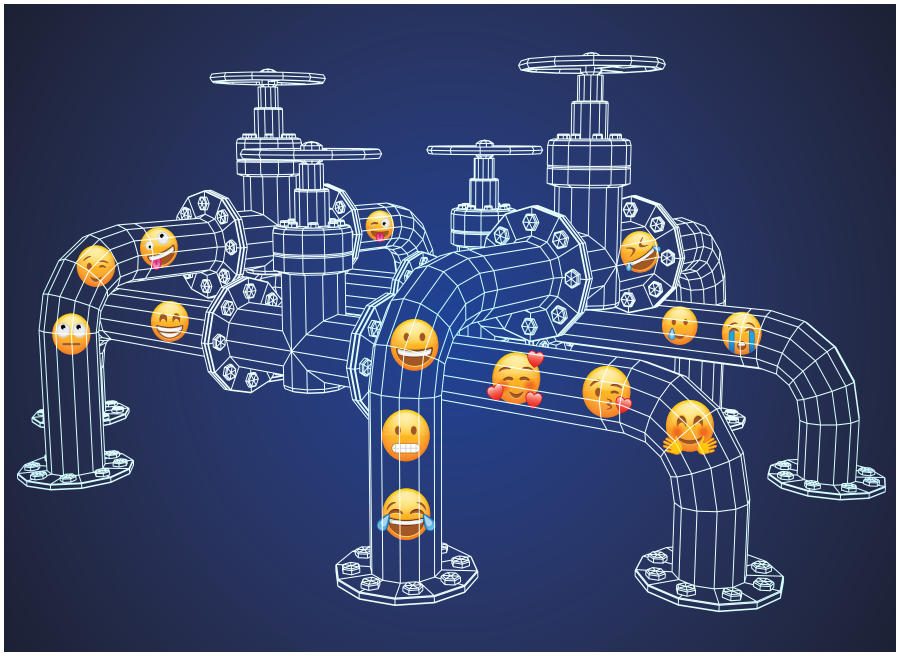


IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK.COM

While social media connects billions, it also fosters discord, with algorithmic amplification of misinformation linked to polarization,[5] violence, and mental health decline.[4] Amid these challenges, a key question is often overlooked: *Who* should govern social media? As social cohesion erodes, the need for democratic oversight grows more urgent. Corporations such as Meta have repeatedly prioritized engagement and ad revenue over the public good, even when aware of the harm their practices cause.[4]

In the U.S., governance is shaped by Section 230 and the First Amendment, limiting state intervention and relying on counterspeech to combat falsehoods.[13] The European Union's Digital Services Act imposes more direct obligations for content moderation.[a] Yet both approaches ultimately defer to corporate discretion, perpetuating a conflict of interest: Misinformation drives user engagement, and engagement drives profits.

This dynamic constitutes a negative externality—private actions imposing societal costs akin to environmental pollution—which justifies corrective intervention to prevent market failure.[14] The "Bullshit Asymmetry Principle" captures the challenge: Refuting misinformation takes significantly more effort than producing it.[15]

Treating social media as a public utility curtails profit incentives for misinformation and enables public oversight of content moderation. Like water, electricity, and broadband, social media is critical public infrastructure, serving as a dominant news channel and forum for societal discourse.[9] Building on prior proposals for public alternatives to corporate platforms,[10,11] we develop a model centered on democratic oversight, transparent moderation, legally bounded content removal, and institutional protections for free expression. While ambitious, this proposal seeks to expand what is politically imaginable by reinforcing the democratic foundations of social media governance. Our framework can accommodate private alternatives under public regulation and is adaptable to diverse democratic contexts. Even though the U.S. exemplifies how partisan government intervention offers no viable path toward replacing commercial freedom with civic responsibility, countries such as Germany, Canada, the U.K., and Australia offer proven non-partisan governance models to draw on.

Adapting the Public Utility Model to Social Media Governance

Governing social media is challenging due to its ever-evolving nature, which de-

a Breton, T. Urgent letter to @elonmusk on #DSA obligations. @ThierryBreton ed., Twitter/X (2023); https://tinyurl.com/mwpynu57, 2023.

EngageCSEdu.org

Peer-reviewed Resources for Engaging Students

EngageCSEdu provides faculty-contributed, peer-reviewed, and ACM DL indexed Open Educational Resources (OERs) for a variety of computer science courses.



engage-csedu.org

acm Association for Computing Machinery

fies established regulatory frameworks. While Internet and public broadcasting regulations offer principles—universality, equitable access, quality standards, and resilient funding—these must be adapted to social media's distinct challenges. Chiefly, effective governance requires institutional safeguards to protect free expression and prevent autocratic control.

Social media platforms, whether public or private, face a trade-off between free expression and civil discourse. People value freedom *of* expression but also freedom *from* harmful expression, posing a dilemma when one person's expression harms another. A state monopoly on truth is incompatible with free democracy.[14] So, how can state-governed social media mitigate negative externalities while upholding democratic opinion forming?

Our proposal distinguishes between content removal and content ranking, with democratically defined and authorized editorial rules. AI can enhance moderation efficiency by detecting illegal content, but auditability and open source algorithms are essential for trust and accountability.

Content removal should be limited to illegal material, enforced by courts. For instance, U.S. First Amendment protections exclude obscenity, child abuse material, incitement to imminent lawless action, true threats, fraud, defamation, and false advertising. Legal misinformation, however, should be addressed through content ranking, not removal, to safeguard free expression.

Content ranking should prioritize social connections and user interests over engagement to curb misinformation. Platforms such as Reddit[12] use upvotes and downvotes to rank content, which can curtail misleading yet technically legal "gray-area" content more effectively than simple flagging.[1] Thus far, countries such as the U.S., U.K., Canada, and Germany regulate only illegal content. While this avoids a state monopoly on truth, it fails to address harmful but lawful expression. Our proposed public utility model combines law-driven removal of illegal content with ranking systems that curb the visibility of harmful material without deleting it. This approach limits its amplification without censorship or government overreach. Decentralized identifiers[b] can prevent vote manipulation while preserving privacy. Self-sovereign identities[c] enable cross-platform data portability, reducing platform dependence.

Any ranking system entails a trade-off between group preferences and societal cohesion. Conservatives may upvote content that liberals downvote. An algorithm marketplace could enhance in-group satisfaction and reduce inter-group conflict, but risks deepening echo chambers. Aggregated preferences may mitigate fragmentation but marginalize outliers. For public social media, prioritizing consensus through aggregation may be preferable. This approach sidelines, but does not censor, extreme content and leaves it accessible to those who seek it. Integrating social graphs and user interests, such a system supports civil discourse while maintaining an open marketplace of ideas.

Checks and balances are essential to prevent partisan capture by insulating content ranking policies and funding decisions from partisan control. The German public broadcasting model offers a valuable precedent. Germany's constitutional court has repeatedly upheld that public broadcasting must be pluralistic and independent. Politicians are barred from controlling editorial or funding decisions. Regulatory boards that determine funding and editorial policy are legally required to include members chosen by designated societal groups (for example, unions and religious organizations), not politicians.

Australia's Online Safety Act 2021

b W3C. Decentralized Identifiers (DIDs) v1.0 (2022); https://www.w3.org/TR/did-1.0/.

c W3C. Identity and the Web (2025); https://www.w3.org/reports/identity-web-impact/.

Any ranking system entails a trade-off between group preferences and societal cohesion.

offers a complementary pathway. It empowers an independent *eSafety Commissioner* to mandate the removal of illegal or clearly abusive material across digital services. While it expands regulatory capacity, it does not address "harmful but lawful" expression or algorithmic ranking. Nonetheless, it demonstrates how liberal democracies could build on existing institutions to move toward a public utility model of social media.

Like public broadcasters, public social media should follow content ranking standards overseen by independent institutions to ensure accuracy, impartiality, and educational value. This may require constitutional amendments to create safeguards against partisan interference. Financial independence can be supported through a mix of fees, donations, and revenues from advertising and premium services.

Public and private social media platforms can coexist, much like private TV stations do alongside public broadcasters like the BBC. However, private platforms must be subject to regulatory oversight to ensure responsible ranking and curb engagement-driven harms. Independent regulators—modeled after the U.K.'s Ofcom—could assess compliance using metrics such as *attention allocation* (how algorithms allocate user attention), *revenue attribution* (how attention is monetized), and *impact measurement* (societal consequences of content amplification).[8] Regulatory tools could range from attention trading mechanisms[14] to more forceful measures such as expropriation, antitrust action, or shutdowns in cases of persistent non-compliance.

Another approach—fully compatible with our proposal—would mandate a shared infrastructure supporting multiple ranking systems.[13] Bluesky's *marketplace of algorithms*[d] is one such model, allowing users to choose their preferred ranking systems. A default public-interest algorithm could coexist with commercial alternatives, provided they meet regulatory standards for misinformation control.[14] However, having a multitude of competing social media infrastructures, including a public one, would be preferable, as a unified infrastructure would introduce a single point of failure and make social media as a whole more vulnerable to partisan takeover. Having a public option alongside private social media would be best suited for supporting a pluralistic and democratic society.

Despite its promise, the public utility model faces challenges.

Weighing Alternative Options

Alternatives to the public utility model abound. In response to mounting calls for accountability, Meta established an external "Oversight Board" in 2020 to review content decisions and adjudicate user appeals. Likened by CEO Zuckerberg to "a Supreme Court,"[7] the board remains structurally and financially dependent on Meta, ultimately leaving policy control in corporate hands. Without accountability beyond Meta itself, the Oversight Board functions more as a PR stunt than a genuine system of checks and balances.[3] By contrast, the public utility model assigns such oversight to *actual* democratic institutions—courts and parliaments—instead of corporate-appointed, pseudo-independent bodies that remain accountable to shareholders rather than citizens.

Recently, Meta shifted from fact-checking moderation to crowdsourced moderation, citing concerns over scalability and the neutrality of fact-checking organizations. However, this shift sidesteps the core issue: Meta retains both the incentive to promote engagement-boosting misinformation and unilateral control over content ranking and removal. Crowdsourcing may broaden participation in moderation, but it does not resolve structural power asymmetries or the absence of public oversight.

Another proposal, *information fiduciaries*, would impose legal duties of care on platforms.[2] Critics[6] caution that this model requires platforms to balance the conflicting interests of shareholders and society. Because engagement simultaneously drives profit and societal harm, fiduciary platforms may struggle to navigate this conflict of interest, limiting their effectiveness. The public utility model offers a stronger solution by removing profit incentives and aligning operations with societal goals such as equitable access, content integrity, and harm reduction. It also ensures democratic oversight by courts and legislatures, with international cooperation addressing the longstanding regulatory evasion by global corporate platforms.

Recent efforts by President Trump to defund PBS and stack its board with political allies highlight the risk of partisan capture,[e] yet they also showcase the relative resilience of democratic institutions: PBS has retained editorial independence, supported by citizen donations and legal protections. In contrast, Meta's content policies rest entirely on executive discretion, requiring no approval beyond Zuckerberg. While imperfect, democratic governance may offer stronger safeguards than corporate self-regulation.

Another proposal envisions a wave of smaller, self-organizing platforms enabled by targeted regulation to address misinformation.[16] While promising, profit-driven smaller platforms funded by advertising would still prioritize engagement, and thus retain incentives to amplify misinformation. A public utility model may address this more effectively by aligning incentives with the public interest.

The Road Ahead

Despite its promise, the public utility model faces challenges. More than anything, social media's resilience to capture depends on institutional safeguards rather than on technical design. In robust democracies, public oversight can reinforce accountability and make capture costlier, warranting serious experimentation with the public utility model. Funding may require taxes—a trade-off justified by their contribution

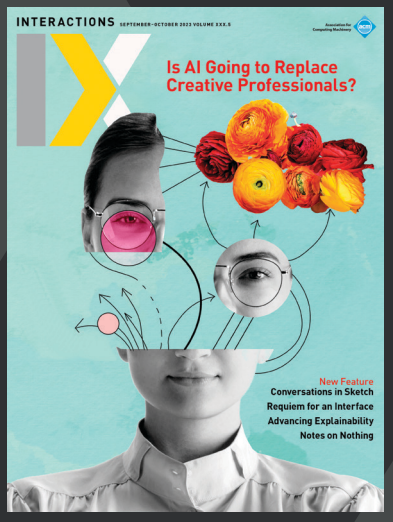
d Graber, K. Algorithmic choice (2023); https://bsky.social/about/blog/3-30-2023-algorithmic-choice.

e See https://theconversation.com/trump-is-aiming-to-silence-public-media-in-the-us-and-if-he-succeeds-his-supporters-here-will-take-note-260584.

INTERACTIONS

ACM's *Interactions* magazine explores critical relationships between people and technology, showcasing emerging innovations and industry leaders from around the world across important applications of design thinking and the broadening field of interaction design.

Our readers represent a growing community of practice that is of increasing and vital global importance.



To learn more about us, visit our award-winning website
http://interactions.acm.org

Follow us on Facebook and Twitter

To subscribe:
http://www.acm.org/subscribe

Association for Computing Machinery

to a democratic society. Reconciling national autonomy with global standards remains complex.

Safeguarding free expression and privacy requires robust institutions resilient to partisan or corporate capture. Such capture occurs in many forms: state control, as seen in VKontakte in Russia, WeChat (Weixin) in China, or attempts of the U.S. government to limit enforcement of the European Union's Digital Services Act in Europe, and corporate influence, as evident in the algorithmic moderation and platform bans shaping discourse in Western democracies. These are not opposite extremes but variations of the same problem: concentrated power over online speech. Private platforms have also been weaponized by governments, such as during the Duterte administration in the Philippines, while public ones may be less likely to amplify coordinated misinformation and less vulnerable to interference by foreign governments. Introducing public platforms alongside private ones could diversify the social media landscape, making it less corruptible overall.

Just as today's private platforms are fragmented, so too might public alternatives fragment along jurisdictional lines if they lack interoperability, weakening network effects. Countries will understandably seek to retain editorial control and fiscal sovereignty. Content removal laws vary widely: Holocaust denial is illegal in Germany but protected in the U.S., while abortion service ads are restricted in some U.S. states but permitted in others. A global enforcement framework, akin to SWIFT, could enable cross-border functionality within domestic legal constraints. Without such coordination, divergent regulations may erode cohesion and effectiveness.

Conclusion

While politically contentious, governing social media as a public utility offers a viable path toward a more equitable and resilient digital society. Drawing lessons from robust democracies (such as the U.K., Germany, Canada, and Australia), we propose a model rooted in non-partisan democratic oversight, law-bounded content removal, and transparent moderation. Although the model alone does not guarantee resistance to partisan capture, particularly if constitutional safeguards are lacking, it does resonate with values shared across ideological lines: free expression, checks and balances, rule of law, sovereign infrastructure, and public accountability. Its purpose is not to empower any party or leader, but to strengthen democratic institutions with sovereign digital infrastructure. If social media is to serve the people, it must be accountable to the people. C

References

1. Allen, J. et al. Quantifying the impact of misinformation and vaccine-skeptical content on Facebook. *Science 384*, 6699eadk3451 (2024).
2. Balkin, J.M. Information fiduciaries and the first amendment. *UC Davis Law Rev. 49*, 1183 (2016).
3. Bentley, J. Meta-funded regulator for AI disinformation on Meta's platform comes under fire: 'You are not any sort of check and balance, you are merely a bit of PR spin'. *PCGamer* (2024); https://www.pcgamer.com/software/ai/meta-funded-regulator-for-ai-disinformation-on-metas-platform-comes-under-fire-you-are-not-any-sort-of-check-and-balance-you-are-merely-a-bit-of-pr-spin/.
4. Horwitz, J. The Facebook Files: A Wall Street Journal Investigation. *The Wall Street J.* (2021); https://tinyurl.com/ycytatjd.
5. Jenke, L. Affective polarization and misinformation belief. *Political Behavior 46*, 2 (2024), 825–884.
6. Khan, L.M. and Pozen, D.E. A skeptical view of information fiduciaries. *Harvard Law Rev. 133*, 2 (2019), 497–541.
7. Klein, E. and Zuckerberg, M. Facebook's hardest year, and what comes next. *Vox* (2018); https://www.vox.com/2018/4/2/17185052/mark-zuckerberg-facebook-interview-fake-news-botscambridge.
8. O'Reilly, T. et al. Algorithmic attention rents: A theory of digital platform market power. *Data and Policy 6*, e6 (2024).
9. Pew Research Center. *Social Media and News Fact Sheet* (2024); https://www.pewresearch.org/journalism/fact-sheet/social-media-and-news-fact-sheet/.
10. Pickard, V. Democratizing the platforms: Promises and perils of public utility regulation. *Media Development 68*, 3 (2022), 6–10.
11. Rahman, K.S. Regulating informational infrastructure: Internet platforms as the new public utilities. *Georgetown Law and Technology Rev. 2*, 2 (2018), 234–251.
12. Roose, K. Reddit's I.P.O. is a content moderation success story. *The New York Times* (2024); https://www.nytimes.com/2024/03/21/technology/reddit-ipo-public-content-moderation.html.
13. Van Alstyne, M.W. et al. Improving Section 230, preserving democracy, and protecting free speech. *Commun. ACM 66*, 4 (Apr. 2023), 26–28.
14. Van Alstyne, M.W. Free speech, platforms and the fake news problem. *SSRN Electronic J.* (2023).
15. Williamson, P. Take the time and effort to correct misinformation. *Nature 540*, 7632 (2016), 171–171.
16. Zuckerman, E. The case for digital public infrastructure. *Knight First Amendment Institute at Columbia University* (2020).

Christoph Mueller-Bloch (muellerbloch@essec.edu) is an assistant professor at ESSEC Business School, Cergy, France.

Raffaele Ciriello (raffaele.ciriello@sydney.edu.au) is a senior lecturer at the University of Sydney, New South Wales, Australia.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3790104

Jiqun Liu

Opinion
Bounded Minds, Generative Machines

Envisioning conversational AI that works with human heuristics and reduces bias risk.



IMAGE BY ANNA_DESIGN/SHUTTERSTOCK.COM

Conversational AI systems are entering the mainstream as a primary interface for information seeking and decision making. Yet most systems, evaluations, and implicit user models still presume idealized users who can carefully weigh evidence and notice implausible claims. In reality, human cognition is bounded by limited time and attention, uneven domain knowledge, and reliance on heuristics that evolved to support fast judgments under uncertainty.[9] Heuristics such as anchoring, availability, confirmation, authority, and scarcity are not failures of reasoning but adaptive mental shortcuts. When paired with fluent, personalized, and confident generative responses, however, these shortcuts become a double-edged mechanism that can systematically skew judgment.

Researchers in human-computer interaction, cognitive science, and behavioral economics have long studied these biases, and there is growing recognition in AI ethics and safety communities as well.[2] However, mainstream design practices and system evaluations still emphasize correctness, relevance, and narrowly defined utility while leaving cognitive impact underexplored.[5] In multi-turn dialogue, a single confident response can anchor users, vivid anecdotes can dominate base rates, repeated agreement can reinforce expectations and misbeliefs, and even neutral layout or ordering choices can introduce framing effects that alter trust and preference.

The risk is not merely that conversational AI fails to mitigate these biases; it is that systems can exploit them for scalable behavioral manipulation, which aligns with the unacceptable-risk framing under the EU AI Act.[a] In advertising, persuasion, and political microtargeting, systems already optimize engagement by aligning with predictable vulnerabilities.[1] As generative models become more adaptive and personalized, subtle steering becomes easier to produce and more difficult to detect.

The challenge, then, is not simply higher factual accuracy. It is to build conversational systems that recognize how users actually reason through heuristics that can be adaptive or distorting depending on context, individual differences, and system choices (see the accompanying figure).

An emerging line of work, sometimes described as machine psychology, studies how generative models exhibit and interact with cognitive shortcuts and how systems might support better reasoning rather than am-

a See https://www.europarl.europa.eu.

plify bias.[8] This motivates a research pathway that connects design, evaluation, and governance to reduce AI harms and bias risks in practice.[6]

Why this becomes a research pathway now? Bounded rationality has long been studied, but its implications become substantially more urgent with large language models (LLMs) and agentic conversational systems being widely deployed. Unlike earlier information systems, conversational AI can adapt natural language turn by turn and influence reasoning over longer horizons, amplifying cumulative effects of framing, anchoring, and heuristic reinforcement. During the next decade, this makes cognitive impact a first-order design concern rather than a peripheral issue.

Designing for Heuristics, Not Against Them

This work does not argue for a single design solution. Instead, it highlights recurring cognitive dynamics that define a broader research pathway for conversational AI as heuristics unfold across multi-turn interaction. Once we recognize that people rely on heuristics rather than mathematically rational utility calculation, the research challenge is to understand how these shortcuts behave when responses are generated fluently, adaptively, and at scale across varying tasks. What once appeared as subtle laboratory effects can accumulate across everyday exchanges, where small shifts in wording, sequencing, and emphasis reshape judgments, comparisons, and preferences.

According to Prospect Theory,[9] people evaluate outcomes relative to reference points and weigh losses more than equal gains. In conversational settings, reference points are often established implicitly through language, ordering, and tone, and early framings can persist across turns. Over time, path-dependent agreement can narrow exploration even without introducing factual errors.

The structure of option sets also reshapes outcomes. When comparisons are generated on the fly, dominated alternatives may appear and quietly steer choices. For example, a student asking an LLM for study resources might see a free open textbook, a reasonably priced course, and an overpriced "platinum package" with little added value; the decoy makes the paid course look like the safe and balanced choice. Similar dynamics arise with anchoring, vivid anecdotes, authoritative tone, scarcity cues, and confirmation-seeking language, which can co-occur and reinforce one another in conversational information retrieval and recommendation.[5]

These dynamics explain why surface-level accuracy cannot be the only standards for judging conversational AI. A response may be correct yet still entrench loss aversion, amplify overconfidence, or channel decisions through hidden framing. The research question shifts from whether an answer is correct to whether an interaction supports sound reasoning within bounded cognitive limits: Does it broaden perspectives, calibrate uncertainty, and reduce vulnerability rather than compound it? Because interactions unfold turn by turn and are context-dependent, this pathway calls for design and evaluation centered on cognitive impact in addition to quality-focused effectiveness.

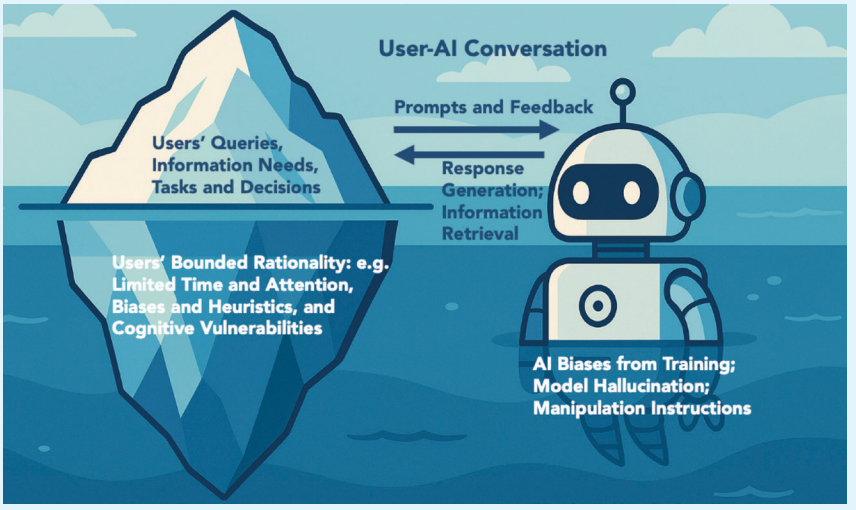
Detecting and Intervening in Bias Risks

Conversations are where reasoning develops between human and machine, and also where it can quietly bend. This pathway requires monitoring how exchanges unfold and identifying signals of bias risk: narrow framing, unbalanced emphasis on benefits versus costs, overconfident claims without uncertainty, comparisons that hide baselines or include redundant options, and rapid user acceptance without examining evidence. Such signals suggest measurable features that future systems could track, including viewpoint diversity via source entropy,[4] calibration failures in expressed probabilities,[3] and turn-level traces that enable later audit.[6] Additional behavioral and physiological signals may help in some settings, but generalizable, privacy-aware signals remain a core research challenge.

Once signals are detected, systems could explore lightweight interventions before presenting a final response: surfacing a clear baseline, framing outcomes symmetrically in gain and loss terms, and representing uncertainty as ranges rather than single numbers.[3] Retrieval can supply base rates and counterexamples.[4] A dominance check can flag or remove decoys, and an anchoring guard can introduce a second estimate after baselines are shown so the first number does not dominate judgment.[10] These interventions can be implemented through structured outputs, function calls, and lightweight critics that review drafts before answers are returned.[7]

Illustrative scenarios can clarify the goal. In health advice, a vivid patient

Challenge in user-AI conversation: System responds to the explicitly issued prompts, but cannot detect underlying bounded rationality or support the implicit human heuristics "under the water."



story presented first can dominate evidence from clinical trials; pairing anecdotes with prevalence rates can rebalance judgment. In e-commerce, a high-priced decoy can make a mid-range product appear reasonable; detecting and labeling dominated options reduces hidden steering. Similar failures can arise in legal guidance and news summarization through redundant options, problematic framing, or anchoring order. These cases show how heuristics can turn from shortcuts into vulnerabilities and why small interventions may keep conversations more balanced under bounded rationality.

Evaluation also has to evolve. Accuracy and topical relevance remain necessary but are no longer sufficient. Systems should be tested on whether interactions improve calibration after uncertainty is shown,[3] broaden perspectives, and reduce susceptibility to dominated options and decoys.[6] Scenario sets that vary framing, priors, and time pressure can stress test interventions, while online experiments can compare policies through stability of choices and reduced repeat queries.[7] Counterfactual logging and interleaving can evaluate alternative baselines safely, and tuning can incorporate penalties when high-risk turns pass without intervention.[6,7]

Open challenges. Detecting cognitive vulnerability in real time is subtle, context-dependent, and heterogeneous across users and domains. Interventions must balance effectiveness with intrusiveness, since excessive safeguards may increase cognitive loads and reduce trust or usability. Integrating detection, intervention, evaluation, and governance into a coherent conversational pipeline raises unresolved questions about transparency and generalizability. These challenges are central to the research pathway rather than solved engineering tasks.

From Principles to Practice

Turning these ideas into practice does not require reinventing conversational AI, but it does require reframing priorities. Many building blocks exist, yet their effective integration and evaluation remain open research problems. Function calls, structured outputs, and long-context modeling can support baseline surfacing and uncertainty representation; retrieval-augmented generation and reasoning can supply diverse perspectives with grounded sources;[4] and lightweight critics can flag missing baselines or overconfident tone.[7] Recent model developments make these capabilities increasingly feasible, but feasibility alone does not guarantee cognitive benefit and safety.[b]

Turning these ideas into practice does not require reinventing conversational AI, but it does require reframing priorities.

Open research questions and roadmap. This pathway suggests several actionable questions: How can systems detect moments of cognitive vulnerability reliably across tasks and populations? Which interventions, such as contrastive framing, baseline surfacing, or uncertainty representation, work best under different heuristics and contexts? How should success be evaluated beyond accuracy, especially in calibration, stability of judgment, and resistance to manipulation? What governance mechanisms and audit trails are needed to make such systems accountable at scale?

Evaluation can be reframed around these goals. Instead of asking only whether an answer is correct, we can ask whether interaction improves calibration, reduces anchoring, and broadens perspectives.[3,6] Scenario sets that vary framing, priors, and time pressure can test robustness, while online experiments can compare policies through stability of choices and reduced susceptibility to decoys.[7] Over time, such measures can complement accuracy and latency with benchmarks for decision quality and cognitive robustness.

Practical safeguards also depend on governance and transparency. Cognitive-impact reviews can become part of release practice alongside privacy and security checks. Policy and model cards can disclose guardrail scope and known limitations, while audit trails can log which options were shown, what baselines were included, and which interventions were triggered.[6] Practical audit trails can include retrieval queries, ranking choices, cited spans, and the system's internal plan behind high-stakes answers.[c] Such documentation may feel burdensome, but it enables reproducible traceability for accountability.

Taken together, these directions outline a credible and timely research pathway for building conversational AI systems and evaluation pipelines that work with bounded human minds, support robust reasoning, and reduce the risk of hidden harm and cognitive behavioral manipulation. C

b See OpenAI, Introducing o3 and o4-mini; https://openai.com/index/introducing-o3-and-o4-mini/.

c See Anthropic Docs, Computer use tool; https://docs.anthropic.com/en/docs/agents-and-tools/tool-use/computer-usetool.

References

1. Carroll, M., Chan, A., Ashton, H., and Krueger, D. Characterizing manipulation from AI systems. In *Proceedings of the 3rd ACM Conf. on Equity and Access in Algorithms, Mechanisms, and Optimization*. (2023); 1–13.
2. Draws, T. et al. A checklist to combat cognitive biases in crowdsourcing. In *Proceedings of the AAAI Conf. on Human Computation and Crowdsourcing 9*, (2021), 48–59.
3. Guo, C., Pleiss, G., Sun, Y., and Weinberger, K.Q. On calibration of modern neural networks. In *Proceedings of the Intern. Conf. on Machine Learning*. PMLR (2017), 1321–1330.
4. Lewis, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems 33*, (2020), 9459–9474.
5. Liu, J. A behavioral economics approach to interactive information retrieval. *The Information Retrieval Series 48* (2023).
6. Magooda, A. et al. A framework for automated measurement of responsible AI harms in generative AI applications. (2023); arXiv:2310.17750.
7. Ouyang, L. et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems 35*, (2022), 27730–27744.
8. Rahwan, I. et al. Machine behaviour. *Nature 568*, 7753 (2019), 477–486.
9. Tversky, A. and Kahneman, D. Advances in prospect theory: Cumulative representation of uncertainty. *J. Risk and Uncertainty 5*, 4 (1992), 297–323.
10. Shunyu Yao, S. et al. React: Synergizing reasoning and acting in language models. In *Proceedings of the Intern. Conf. on Learning Representations (ICLR)*. (2023).

Jiqun Liu (jiqunliu@uwm.edu) is an associate professor at the University of Wisconsin-Milwaukee, Milwaukee, WI, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3789513

James A. Landay and Anoop K. Sinha

Opinion
The Multimodal Agent Interfaces Paradigm Shift

How the movement toward multimodal agent interfaces will fundamentally transform user interactions with digital systems.

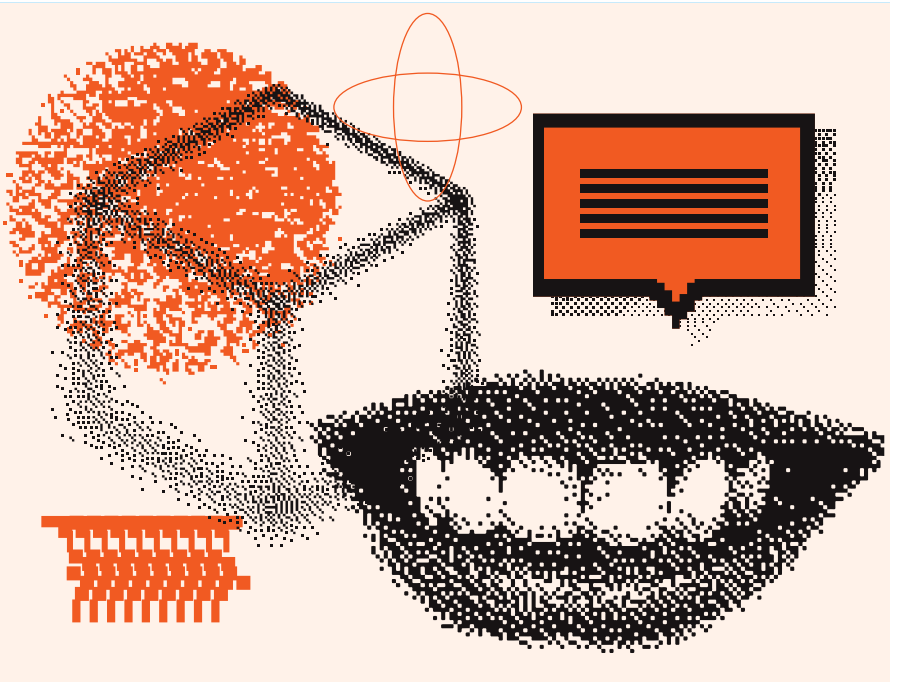


IMAGE BY SOFTULKA

Imagine a world where your devices understand your natural communication—both what you do and what you mean. Most interface innovation up to now has been about what you do. Early graphical user interfaces (GUIs) made computing more intuitive via tool-oriented direct manipulation and WYSIWYG ("What You See Is What You Get"), but still relied on users to translate their goals into specific commands, as with earlier command line interfaces (CLIs). This gap is "the gulf of execution" in the parlance of Don Norman.[5] While GUIs combine output modalities like text, sound, and graphics, their input interaction model remains rooted in discrete commands and direct manipulation.

AI is rapidly increasing computers' ability to better understand what you do, creating a world where you can also communicate with technology explicitly using additional input modalities such as voice (for example, "What will the weather be today?") or touch gesture, or implicitly by what or where you are doing something in the physical world (for example, Apple Maps suggesting where you might want to go next based on where you currently are and what is next in your calendar).

AI is also changing the paradigm of interfaces to one that *understands what you mean* at a higher level of abstraction through more natural communication (for example, "make the colors in this slide deck more vibrant"). Now, with these new capabilities we can have what we could call WIN-WIN ("What I Need When I Need") interfaces, with AI stepping in at the right time to understand our intent, enabling more expressive, natural interactions and correspondingly powerful new functionality.

Imagine an educational platform that uses AI to personalize a learning interface to a student's needs. It analyzes their facial expressions (detecting confusion or engagement—but only after consent by the student and their parents), their intonation (identifying hesitation or confidence), their interaction and learning patterns (understanding their existing mastery, pace, and preferences), along with what they ask for, verbally or with the keyboard, to adjust the learning content, pace, and modalities in real time. Like a good teacher, a learning multimodal agent will know when to give encouragement and when to let the student productively struggle with material.

This is the promise of multimodal agent interfaces (MAIs)—a paradigm shift that puts intentions and natural communication at the center of design. Multimodal agent interfaces are intent-centered systems that allow us-

ers to fluidly express their needs using multiple modalities in substitution or simultaneously across devices, communicate output and disambiguate input similarly across multiple modalities and devices, and anticipate user needs based on past interactions. This interaction evolution is fueled by the convergence of powerful AI, ubiquitous connectivity, and increasingly sophisticated devices, enabling technology to adapt to our natural communication.

Past UI Paradigm Shifts

A paradigm shift in user interfaces signifies a fundamental change in how users interact with computers. Command line interfaces (CLIs) brought real-time user input and output to time-sharing systems and then to microcomputers. The personal computer introduced the mouse and keyboard and GUIs to a broad audience, enabling direct manipulation of digital objects. Mobile interfaces extended GUIs and included basic multimodality, adding new layers of mobility and immediacy. For example, the iPhone touchscreen went beyond the limits of pen-based tablets of the time by using "the best pointing device in the world" (the finger) and a dynamic on-screen keyboard to access the entire Internet through a standard web browser. Something happening in each of those shifts is an increase in the *bandwidth* of communication between the user and the computer—the amount of information that can be transmitted per unit of time.

This evolution continues with MAIs, which increase the bandwidth further by centering the interaction on user intent rather than a specific command or direct manipulation. This increased syntactic bandwidth (for example, an image with a large amount of data) and semantic bandwidth (for example, a high-level written or spoken prompt specifying the user's intent) supports more natural communication, allowing the user and their goals to increasingly be understood by the computer in the same way they might be *understood* by another person, reducing the gulf of execution. The transition from GUIs to MAIs is thus a shift in focus from mastering physical manipulation to expressing conceptual intent.

A paradigm shift in user interfaces signifies a fundamental change in how users interact with computers.

However, for a "pointing" task, an AI chat interface such as ChatGPT or Gemini might be lower bandwidth than a finger on an iPad if describing the location is difficult (for example, "the left, lower corner of the red rectangle"). This illustrates the importance of multimodal interfaces that allow users to use the best modality, in this case direct manipulation, for the task they are trying to achieve. But higher level or more complex tasks that can be more easily described using language (for example, "move all the larger red boxes on the first 10 slides left by 5 pixels") will often offer higher communication bandwidth.

The AI Paradigm Shift Underway

After ChatGPT was introduced in November 2022, the world saw the power of large language models and their ability to interact in a natural, conversational manner. This sparked a wave—rather, a tsunami—of innovation in the field of AI, with new models and applications advancing rapidly.

Let's take ChatGPT Voice Mode and Gemini Live, for example. They provide sophisticated interaction including streaming camera and voice input, allowing users to converse with the AI model using spoken language and real-time video of their current context, making the interaction feel more fluid.

Even more prevalent is the ability to express intent with typed natural language prompts, which is already becoming a central interaction mode. Code generation tools like GitHub Copilot and Cursor, for example, can help developers write code more efficiently by suggesting code snippets, identifying potential errors, and even generating entire functions based on natural language descriptions. "Vibe coding" takes this one step further with non-programmers creating entire applications based on a specification written in a series of prompts.[6] How does that change the way we need to think about design processes?

These advancements, happening across AI applications simultaneously, are increasing the overall bandwidth of communication between humans and computers. These new interfaces are a major step toward intent-based interactions.[4]

History of the Multimodal Agent Interfaces Paradigm

The core components or primitives of MAIs—agents, multimodal input/output, and contextual interaction—all have long histories and all have been making rapid advances recently.

Agents. The concept of software agents[a] gained prominence in the 1990s as part of AI research, with early examples such as Apple's 1987 Knowledge Navigator concept video and Microsoft's Clippy 10 years later. These agents were designed to assist users by performing tasks autonomously or semi-autonomously, a vision inspired by Alan Turing's 1950 work on machine intelligence and John McCarthy's idea of programs reasoning about goals.[2] The field advanced significantly with the rise of machine learning. Today's agents demonstrate capabilities in reasoning, context-awareness, and adaptive personalization, moving us toward AI collaboration with users.

Multimodal interfaces. Multimodal interfaces, which integrate multiple input methods (for example, speech, gesture, and touch) and, often, multiple outputs (for example, audio and graphics), began gaining traction in the research community in the late

a Note that we use the term "software agent" in a more specific and interactive way than the more general definition of software agent as "a computer program that acts for a user or another program in a relationship of agency" (see https://en.wikipedia.org/wiki/Software_agent). In our usage, a user communicates with an agent about what they want and the agent carries it out for or with the user and communicates the result. The multimodal interface is the communication part between the user and the agent for both the input and output portions of that interaction.

1980s and early 1990s. The seminal work "Put That There" by Bolt and colleagues introduced the concept of combining speech and gesture for interaction in interactive environments.[1] This research laid the foundation for systems that could interpret multiple modalities. The development of deep learning models, such as CLIP (OpenAI) and Flamingo (DeepMind), has enabled modern multimodal systems to process and align diverse input types simultaneously, leading to breakthroughs in advanced conversational agents.

Contextual interaction. The idea of contextual interaction, where systems adapt based on environmental and user context, traces back to the 1990s with Mark Weiser's famous vision of ubiquitous computing.[7] Weiser proposed that computers should "weave themselves into the fabric of everyday life" by becoming aware of their surroundings and users' needs. Recent advancements in deep learning and sensor fusion, seen in devices like smartwatches, allow systems to infer simple forms of user intent and adapt dynamically. Contextual interaction will play a central role in future multimodal agent interfaces, bringing together implicit and explicit user inputs.

The Rising Multimodal Agent Interfaces Paradigm

Imagine a context-aware fitness tracker that is more akin to a coach: this fitness tracker goes beyond simply counting steps or recording a run. It integrates data from your wearable sensors (heart rate and activity levels), your smartphone's GPS (location and terrain), the current weather, your long-term goals and the barriers to exercising you've expressed previously, and even your voice input ("I'm feeling tired") to provide real-time feedback and adjust your workout plan dynamically. Just like a professional coach, it might suggest taking a break, changing your route, trying a different exercise based on your current context, or pushing you to continue what you have started.

This new era raises several profound questions:

- How do we design MAIs that are both powerful and intuitive, capable of understanding our intentions in complex, ever-changing environments?
- How do we successfully infer intent by fusing the explicit interactions of users (for example, multimodal inputs) with their implicit interactions (for example, the user's current context, history of use, and knowledge)?
- How do we ensure these MAIs remain trustworthy and transparent, addressing both the potentially widening gulf of execution[5] and the new process gulf[3] that emerges from autonomous, proactive, intelligent systems?

The multimodal agent interfaces paradigm shift is worth paying close attention to.

The framework this suggests is that interactions can be powered by both explicit and implicit input from users, or often, combinations thereof. With AI-driven interaction moving beyond simple task completion to more contextual understanding and proactive assistance, the answer to these questions lies in multimodal agent interfaces that seamlessly facilitate anytime, anywhere interaction. A still open and important research question for these user interfaces that adapt on their own is how to make the adaptations predictable so that users are able to find the functionality they are looking for.

The key shift is from reactive interfaces that present fixed options to proactive ones that actively help solve problems and suggest approaches *collaboratively*. But this raises important questions about user agency and the balance between automation, user control, predictability, and privacy. Current trends suggest we are moving toward a hybrid model where AI handles routine tasks while leaving strategic decisions and creative direction to humans, though the right balance in different domains and with different user populations will require research exploring and evaluating these tradeoffs, including the critical impact on accessibility for diverse populations.

Conclusion

The multimodal agent interfaces paradigm shift is worth paying close attention to. This evolution represents a fundamental change in HCI, shifting the core interaction model from mastering commands and physical manipulation to expressing conceptual intent. As AI enables us to finally narrow the long-standing gulf of execution, it creates new possibilities for more natural and expressive interaction. However, this transformation introduces critical challenges that future research must address, namely, how to design systems that maintain trust, transparency, and user agency, while avoiding a new process gulf and ensuring accessibility for all populations. Just as businesses that clung to text-based interfaces became obsolete, organizations that ignore this shift risk falling behind as users increasingly expect more intelligent, responsive, and natural ways of interacting with computing. C

References

1. Bolt, R.A. "Put-that-there": Voice and gesture at the graphics interface. In *Proceedings of the 7th Annual Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH '80)* (July 1980), 262–270; 10.1145/800250.807503.
2. McCarthy, J. *Programs with Common Sense* (1959); https://www-formal.stanford.edu/jmc/mcc59.html
3. Morris, M.R. HCI for AGI. *Interactions 32*, 2 (2025), 26–32; 10.1145/3708815.
4. Nielsen, J. *AI: First New UI Paradigm in 60 Years.* Nielsen Norman Group (2023); https://www.nngroup.com/articles/ai-paradigm.
5. Norman, D.A. and Draper, S.W., Eds. *User Centered System Design: New Perspectives on Human-Computer Interaction.* CRC Press (1986); https://www.nngroup.com/books/user-centered-system-design/.
6. Roose, K. Not a coder? With A.I., just having an idea can be enough. *The New York Times* (Feb. 27, 2025); https://www.nytimes.com/2025/02/27/technology/personaltech/vibecoding-ai-software-programming.html.
7. Weiser, M. Some computer science issues in ubiquitous computing. *Commun. ACM 36*, 7 (July 1993), 75–84; 10.1145/159544.159617.

James A. Landay (landay@cs.stanford.edu) is a professor at Stanford University, Stanford, CA, USA, and a visiting faculty researcher at Google, Paradigms of Intelligence, Mountain View, CA, USA.

Anoop K. Sinha (anoopsinha@google.com) is a research director at Google, Paradigms of Intelligence, Mountain View, CA, USA.

We thank numerous colleagues for valuable comments and feedback on these ideas, particularly Meredith Ringel Morris, Michael Terry, Jimmy Lin, Blaise Agüera y Arcas, and James Manyika.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3834783

Jeanna Matthews

Opinion

ACM as a Professional Society: The ACM Volunteer Database

Expanding ACM's Volunteer Database could help members better document their varied volunteer service to the organization.



IMAGE BY NSTAFEEVA/SHUTTERSTOCK.COM

I HAVE BEEN CO-CHAIR of the *Communications* Opinion section for more than four years. As I write this, Chiara Renso and I are transferring this role to new leadership: Vivek Sarkar is taking the reins. In our time as co-chairs, I am told that the Opinion section has grown to receive more submissions than the entire magazine did before we started. It was a big job, but we have loved serving, especially together.

When I first started, I wrote the Opinion piece "Embracing Critical Voices" to express what I hoped to accomplish. As I wrap up my responsibilities as co-chair, I am writing a series of columns expressing my hopes for ACM as a professional society, informed not only by my role here in *Communications*, but as co-chair of the Global Technology Policy Council, former chair of the SIG Governance Board, and one of the longest-serving members of ACM Council.

As *Communications* Editor-in-Chief Jim Larus expressed in the recent Opinion piece "Wither ACM? Publish and Perish?," ACM seems to vacillate between being a publisher and a professional society. Over the years, in the context of arguing for Open Access, I have repeatedly argued that "Whenever there is a conflict between being a publisher and a professional society, ACM should be a professional society every day and twice on Sunday." Why? Very simply, every good thing ACM has to offer the world is the direct result of authors, volunteers, and community members. The staff and volunteers in ACM publishing serve ACM members in key ways, amplifying their influence, highlighting their contributions; but to ever be a publisher in opposition to the will of members would be to choke the only true source of value ACM brings.

I agree with Larus that it is "time for ACM members to debate what kind of organization ACM should be and how to remake it into the society they want to belong to." I hope to spark conversation around specific ideas and recommendations for change. Many of my recommendations boil down to simple principles—do not surprise members, involve members substantively in the big decisions facing ACM, create processes that respond to member requests, and help members organize around areas of shared concern.

In this first column, I would like to talk about publicly celebrating members' volunteer service in an expanded Volunteer DB.

In June 2022, as a member of ACM Council, I suggested that ACM create a Volunteer Database to connect volunteer leaders with other people who have served in the same roles. The

original goal was to increase the continuity of volunteer leadership. Without this, one generation of volunteer leadership may push for change; then, the next generation comes in fresh without awareness of this history and starts from ground zero to discover the problems for themselves and push for similar change. With volunteer leadership terms relatively short, it is possible that no generation accomplishes the changes. Former *Communications* Editor-in-Chief and columnist Moshe Vardi discussed some of these concerns in his Opinion article, "The Agency Trilemma and ACM."

I am happy to say that the existing Volunteer DB has already made it easier for volunteers to sustain the push for key community initiatives such as the transition to Open Access, but it could be so much more. The current Volunteer Database contains only a small subset of ACM volunteers, and many important roles are missing.

The ACM Digital Library is a comprehensive record of author contributions and similarly, the Volunteer DB could be a comprehensive record of volunteer service. It could provide an authoritative source members could use to document their professional service to share in the context of searching for a job or documenting impact for tenure, immigration, or other purposes.

The existing Volunteer DB has already made it easier for volunteers to sustain the push for key community initiatives such as the transition to Open Access, but it could be so much more.

Have you volunteered for ACM as a reviewer, distinguished speaker, program committee member, program chair, general chair, editor? Look yourself up in the Volunteer Database. Is your record complete and accurate? I'd love to know. If yes, great! If not, one way for ACM to serve members well as a professional society would be to receive reports of missing service and work diligently to improve the database. In time, the Volunteer DB could remind you of roles you forgot rather than the other way around.

There may be problems documenting historical service with sufficient evidence to confidently add a record. That is understandable, but going forward ACM could keep better records documenting and celebrating the varied volunteer service of its members, helping members document and brag about their important work.

Jeanna Matthews (jnm@clarkson.edu) is a professor of computer science at Clarkson University, Potsdam, NY, USA.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs International 4.0 License.

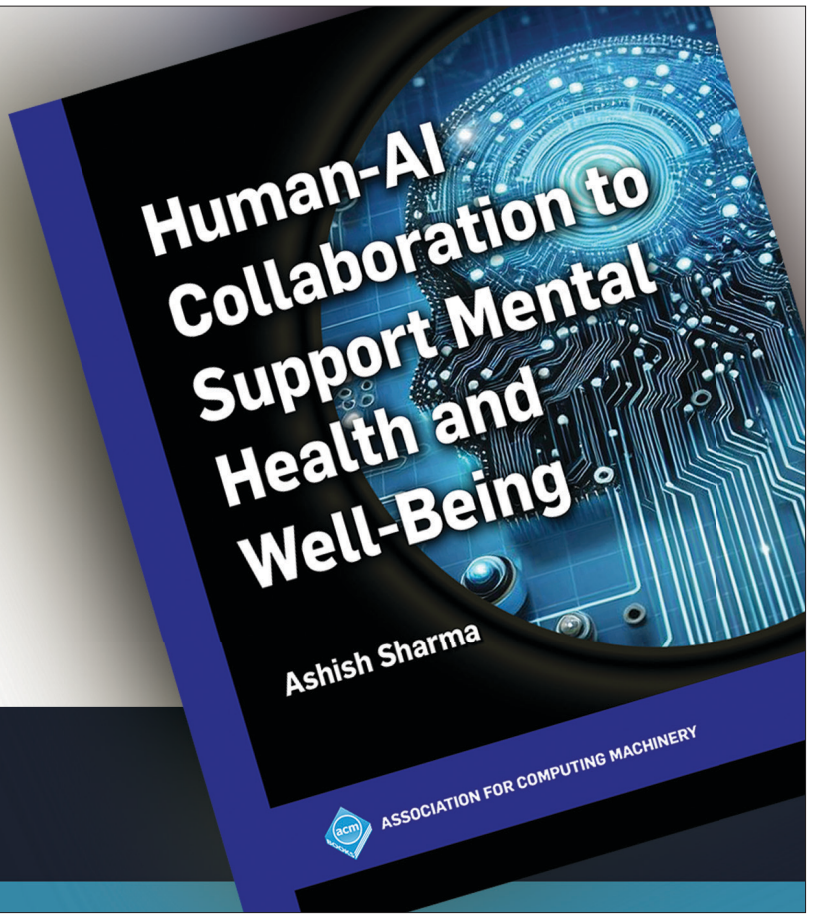
© 2026 Copyright held by the owner/author(s).

Human-AI Collaboration to Support Mental Health and Well-Being

Ashish Sharma

ISBN: 979-8-4007-3184-6
DOI: 10.1145/3749728

http://books.acm.org



ACM BOOKS
Collection III

DOI:10.1145/3797913

Holger Regenbrecht, Kushani Perera, and Nadia Pantidi

Opinion
Everything Everywhere All-Pervasive Augmented Reality

Designing for and with people:
A proposal for a pervasive augmented reality civics space.

EVERYDAY COMPUTING TECHNOLOGY has made its way from the office desk (PC), to the café (laptop), the pocket of your clothing (smartphone), and finally to your wrist (smartwatch). Your eyes and ears are next in line with smart glasses. Smart glasses are worn like ordinary sunglasses or prescription glasses and "see, hear, and locate" your environment to augment it with text, video, and 3D models, for example. No longer the province of research labs only, Big Tech companies such as Google, Microsoft, Meta, Snap, and Apple are now all working on this increasingly popular technology. So, there just might be something to this particular development.

When you are wearing such smart glasses all the time, we call this pervasive augmented reality (AR). Those pervasive AR glasses will always be on, will augment what you see and hear, will respond to the context and your voice and gestures, and, combined with AI, will take care of any apps to be operated. Pervasive AR will constantly and continuously provide you with information at the right place and the right time.[2] Since the glasses are right in front of your eyes and near your ears, there is no longer a need for a smartphone to be pulled out of your pocket or a smartwatch to be looked at.



IMAGE BY ANASTASIIA SAPOZHNIKOVA

So, what could pervasive AR be useful for? There is the promise of a paradigm shift in how we will communicate, collaborate, work, play, educate, learn, entertain ourselves, and see the world. With this always-on and always-there technology we might make decisions faster and more accurately and explore the world while having all the information we need, accessible without effort.

Pervasive AR can be a social bonus, as you can more quickly start conversations between you and people you do not know so well. For instance, we have almost all been in a situation of talking to a person we have met before but have forgotten their name. With the latest developments, your pervasive AR glasses will subtly overlay that name into your view.

However, the introduction of pervasive AR often comes with undesired consequences, in areas such as

privacy, commercial exploitation, distractions, digital inequalities, and in judging what is true and right.[1] Pervasive AR will subject its users to a constant feed of tailored augmentations, of which the sources are unverified, and the access is user specific. This ability of displaying context- and user-specific content is enabled by cameras, microphones, and other sensors on pervasive AR glasses capturing the world around its users continuously and often intrusively. The overlaid digital information, being within the wearer's field of view, needs to be filtered to avoid information overload and health and safety issues, and the information provided needs to be trusted and not manipulated. To date, little is known about such perceptual effects and ethical consequences.

So, what is fundamentally different when it comes to ethical questions with pervasive AR compared to other emerging technologies? Regardless of the specific glasses and their applications, pervasive AR will alter the way its users see the world. In-place, in-time information will be brought to the user, unprompted, without being incited by the user.[2] This also carries the risk of distorting reality perception in a subtle and unprecedented way. Existing research has often discussed how prolonged AR use can blur the distinction between reality and augmentations, leading to users being misled. Furthermore, continuously wearing AR eyeglasses will potentially lead to health and safety issues not seen with other technologies, such as increased cognitive load, fatigue, eye strain, and potential physical harm from manipulated augmentations.[5] In addition, we already see software applications running on AR glasses for law-enforcement agencies, which demonstrate new issues surrounding privacy and dignity that will certainly develop when those glasses become commonplace and pervasive. Those ethical examples present a need to look more closely at pervasive AR.

Anticipated implications of the so-called metaverse will also impact pervasive AR,[1] such as users being unwillingly subjected to corporate-mandated content. Similarly, pervasive AR further exacerbates the already problematic situation of existing racial and sexist biases due to lack of inclusivity in the development of technologies. This includes research that has specifically focused on identifying social implications of everyday use of AR, often exploring themes of privacy concerns,[3] commonly stemming from the always-on nature of cameras.

Our research uses the power of empirical studies with guiding questions[8] for ethical impact assessment of pervasive AR. We can anticipate future real-world use through technology probes, that is, tangible implementations of potential pervasive AR systems, and empirically test them with people in selected scenarios such as socializing in public.[7] Hence, we can receive intuitive feedback from the public instead of relying purely on the opinions of the designers and engineers of such technology. A reoccurring theme is the issue of asymmetry in the future use of pervasive AR—the divide between the technology haves and have-nots. We have explored this asymmetry in terms of how pervasive AR can transform public settings by augmenting not only the space,[6,7] but, also those sharing the space by showing pervasive AR eyeglass wearers information about their bystanders. Asymmetry is multifaceted and raises the questions of who will be able to afford these eyeglasses and who gets which information? Who is controlling the information? Are there new kinds of power imbalance that will potentially develop here? Are we willing to leave this level of decision making to the Big Tech companies, or should we all try to actively shape the future of pervasive AR? Here, we strongly propose the latter choice.

One of the fundamental questions arising is whether realities of alternative facts will be promoted through the use of pervasive AR. As David Chalmers in his *Reality+* book provokes: Will there be an "Apple Reality, Facebook Reality, and Google Reality"? Recently, Meta released its latest iteration of AR glasses, in collaboration with Ray-Ban. These devices exacerbate privacy concerns, as subtle, spectacle-like AR devices with always-on cameras are increasingly thrust upon societies.[3,4] However, it is not only Meta; Samsung, Apple, Google, Snap, and Alibaba have enthusiastically announced their devices will be the new future. That makes one wonder about Chalmers' proposition on the emergence of developer-driven technologies: Are the users at the mercy of the developers? After all, shouldn't users' concerns count for more than another pixel upgrade or another boost to a company's bottom line?

The million-dollar question is how

Combining augmented reality with artificial intelligence as an everyday reality.



PHOTO BY HOLGER REGENBRECHT

to ethically shape the future of pervasive AR. Moshe Y. Vardi[9] has, on multiple occasions, argued that technology should serve humanity, that Big Tech is becoming too powerful and its power must be curtailed, and that its business models are unethical and must be regulated, finally asking the question of whether it is even outright unethical to work for Big Tech.

As a way out of this ongoing dilemma, we propose a co-responsibility model where computing professionals adopt a position of supporting the public good and genuinely co-designing pervasive AR. Such a model allows computing professionals to participate in the future-shaping process without having to either shoulder the whole responsibility for all ramifications or retreat to a position of ethical neglect.

We propose what we call a pervasive AR civics space: a participatory platform to design pervasive AR technology with and for people, potentially leading to high usability and utility, transparency, and meaningful regulation. We have seen how participatory design in the 1970s and 1980s shifted the balance of power between workers and managers by advocating for workers having the ability to influence the design of the computer systems they were expected to use. The foundations of participatory design were, and still remain, greatly political, supporting the empowerment of vulnerable and marginalized groups, and rooted in democratic values; the right of people to co-determine and be active agents of their living conditions. Within this frame, a participatory decision space would leverage principles and practices from digital civics[10] and networked participatory publics to advocate for pervasive AR users to be able to influence the design of, or better control, their immersion at every layer of manipulation. This space should be used to rethink existing divides between the haves and have-nots, between technology providers and users, and between regulators and citizens to enter into a constructive dialogue about the future of this potentially omnipresent and transformative technology.

The million-dollar question is how to ethically shape the future of pervasive AR.

However, given pervasive AR is already in place as part of our socio-technical infrastructure, and 'publics' are heterogeneous and in conflict, this is not an easy task. It may require a participatory design strategy, where practices and technologies undergo change at the same time, and therefore require continuous alignment of actors' conflicting interests and matters of concern. The exact methods to be used are still a matter of debate, but there are existing approaches that can serve as starting points or used in localized contexts, such as Speculative Design or the Living Labs concept that has been successfully applied in contexts such as rural health solutions, urban tourism, and others to support co-design and open innovation. Whatever the method, it will certainly require a multi- and interdisciplinary approach, between technologists, designers, social scientists, businesses, regulators, and of course end users and other stakeholders.

A truly participatory process can be initiated, implemented, and maintained at any type and level of pervasive AR research, design, and development. It allows leaders, line managers, interface designers, software developers, and all other professionals involved in decision making, big or small, to influence the way we will be surrounded by and see the world through pervasive AR. Co-responsibility and initiative have to be developed inside, with, as well as outside of Big Tech. Based on our experience, moral intuition and explicit morality of non-experts and experts alike play a major role here.

A civics space for pervasive AR with computing professionalism at its core can interrogate power relations and changing hierarchies and at the same time bring participative and dialogue-based processes to the forefront of discussions about how such technologies are developed, implemented, regulated, and held accountable. We need informed digital citizens to engage with those processes to confidently welcome pervasive AR to our everyday lives—let's shape smart glasses for a smart future. C

References

1. Eghtebas, C., Klinker, G., Boll, S., and Koelle, M. Co-speculating on dark scenarios and unintended consequences of a ubiquitous(ly) augmented reality. In *Proceedings of the 2023 ACM Designing Interactive Systems Conf.* (July 2023); 2392–2407; doi: 10.1145/3563657.3596073.
2. Grubert, J., Langlotz, T., Zollmann, S., and Regenbrecht, H. Towards pervasive augmented reality: Context-awareness in augmented reality. *IEEE Trans. Visual Computer Graphics 23*, 6 (2017), 1706–1724; doi: 10.1109/TVCG.2016.2543720.
3. Koelle, M., Kranz, M., and Möller, A. Don't look at me that way! Understanding user attitudes towards data glasses usage. *MobileHCI '15: Proceedings of the 17th Intern. Conf. on Human-Computer Interaction with Mobile Devices and Services* (2015), 362–372; doi: 10.1145/2785830.2785842.
4. Koelle, M., Wallbaum, T., Heuten, W., and Boll, S. Evaluating a wearable camera's social acceptability in-the-wild. In *Proceedings of CHI EA '19* (2019), 1–6; doi: 10.1145/3290607.3312837.
5. O'Hagan, J. et al. Augmenting people, places, and media: The societal harms posed by everyday augmented reality, and the case for perceptual human rights. In *Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia (MUM '23)*, 225–235; https://doi.org/10.1145/3626705.3627782.
6. Perera, K., Langlotz, T., Pantidi, N., and Regenbrecht, H. What you see is (not necessarily) what I see—Pervasive AR for public displays (Nov. 2024), 59–73; doi: 10.1145/3726986.3727005.
7. Regenbrecht, H., Knott, A., Ferreira, J., and Pantidi, N. To see and be seen—Perceived ethics and acceptability of pervasive augmented reality. *IEEE Access 12*, (2024), 32618–32636; doi: 10.1109/ACCESS.2024.3366228.
8. Stahl, B., Timmermans, J., and Flick, C. Ethics of emerging information and communication technologies: On the implementation of responsible research and innovation. *Science Public Policy 44*, 3 (June 2027), 369–381; doi: 10.1093/SCIPOL/SCW069.
9. Vardi, M. I was wrong about the ethics crisis. *Commun. ACM 68*, 1 (Jan. 2025), 5; doi: 10.1145/3704959.
10. McCarthy, J. and Wright, P. *Taking [A]Part: The Politics and Aesthetics of Participation in Experience-Centered Design*. The MIT Press (2015), 143–146.

Holger Regenbrecht (holger.regenbrecht@otago.ac.nz) is a professor and the co-leader of the Human-Computer Interaction Lab at the School of Computing at the University Otago, Aotearoa, New Zealand.

Kushani Perera (kushani.perera@otago.ac.nz) is a researcher at the University of Otago, Aotearoa, New Zealand.

Nadia Pantidi (nadia.pantidi@vuw.ac.nz) is a senior lecturer and the current associate dean for research in the Faculty of Architecture and Design Innovation at Victoria University of Wellington, Aotearoa, New Zealand.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3822412

An analysis of how Google migrated a multibillion-line codebase from x86 to Arm.

BY ERIC CHRISTOPHER, KEVIN CROSSAN, WOLFF DOBSON, CHRIS KENNELLY, CANNADA LEWIS, KUN LIN, MARTIN MAAS, PARTHASARATHY RANGANATHAN, EMMA RAPATI, AND BRIAN YANG

Instruction Set Migration at Warehouse Scale

MIGRATING LARGE CODEBASES to a new instruction set architecture (ISA) is a major engineering challenge. Examples include Apple's migration from PowerPC to x86 and later to Arm,[3] as well as the adoption of Arm by major hyperscalers (such as Amazon, Google, and Microsoft). While there are anecdotal claims regarding the complexity and efforts required[1,26,28] and several historical records of such migrations,[23,33,34] to our knowledge, there is no systematic analysis of what these ISA migrations entail *today* and how they are impacted by modern technologies, such as improved software-engineering tools and AI. In this article, we perform such a systematic analysis for the migration of a multibillion-line codebase from x86 to Arm at Google.

When source code is available, as is often the case via open source or within large organizations, recompilation is preferred over techniques such as binary translation.[a] Contemporary ISAs are generally well-supported in upstream compilers, runtime libraries, and the Linux kernel. As a result, modern compilers mostly "just work" for a new ISA, and previous ISA additions have smoothed the path to packages supporting cross-compilation by default. For example, 98% of Debian packages build for RISC-V, although it only became an official Debian architecture in 2023.[9]

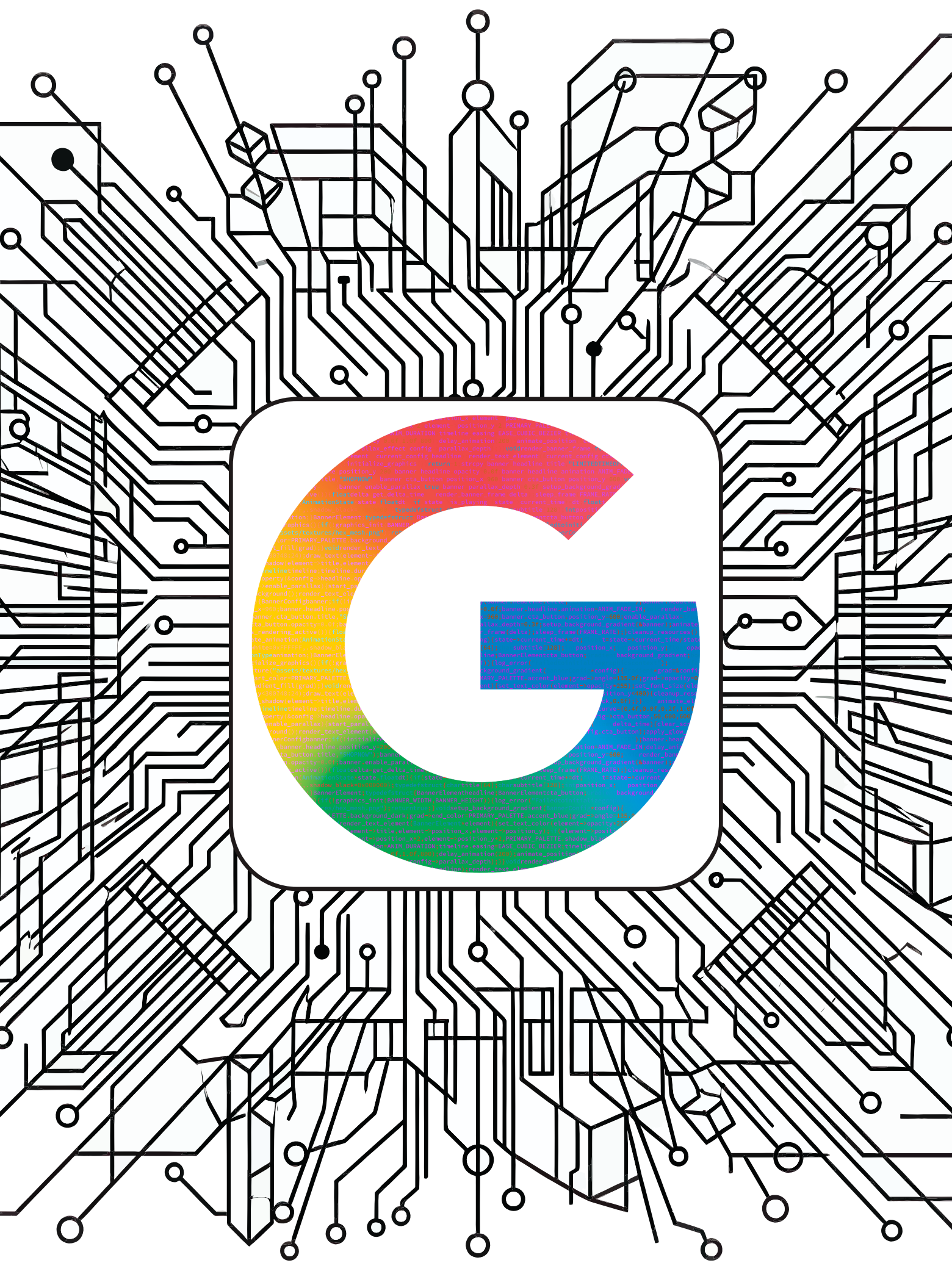
Perhaps surprisingly, this does not mean ISA migration is no longer a challenge in the presence of source code. We find that modern ISA migration involves many usually simple, repetitive tasks, such as updating build scripts or fixing floating-point issues. In contrast to historical migrations, the volume of simple changes is much higher today. This creates challenges but also presents opportunities for automation, including with AI. We focus on three research questions:

1. What are the tasks involved in a modern ISA migration?
2. Which tasks can be successfully automated with conventional methods?
3. How can modern AI help automation, and what tasks remain challenging?

To answer these questions, we provide what we believe is a first-ever detailed breakdown and taxonomy of large-scale ISA migration tasks. Using state-of-the-art LLMs, we analyze and

a This insight dates back to at least the 1990s, as found in the work by Digital Equipment Corp.,[10] for OpenVMS VAX to Alpha from 1996.

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK.COM



categorize a corpus of 38,156 commits that constitute our real-world migration. We also quantitatively evaluate the capability of current tools, including AI models, to perform these tasks automatically, systematically identifying the strengths and weaknesses of current automated tools and highlighting areas of future work and improvement. We believe this work highlights research opportunities for the academic community and revisits longstanding assumptions around ISA migrations.

Specifically, we contribute the following insights:

- The complexity of ISA migrations involves a number of different tasks, many related to rewriting build and configuration files.
- Many of these tasks are highly automatable.
- Many of the tasks that are not automatable only need to be performed once when going from a single ISA to multiarch.
- Of the remaining tasks, many can be performed by modern AI, but some challenges remain.

Background and Related Work

There are several reasons why organizations have performed large-scale ISA migrations. First, many ISAs have gone extinct over the years (e.g., Alpha, MIPS, SPARC, Itanium, VAX). Second, the proliferation of Android and iOS apps drove many codebases to be ported to Arm. Third, Apple Macs went through successive migrations from PowerPC to x86 to Arm. Finally, major cloud hyperscalers have also been migrating large codebases from x86 to Arm.

In this context, migration not only refers to getting software to build on a new architecture, but also to reaching parity in terms of performance, security, and stability. While ISA migrations go back more than 30 to 40 years,[23,33,34] modern migrations are vastly larger and more dependency-heavy than these historical examples. The most closely related academic work falls into two categories.

First, there is a significant amount of work on static and dynamic binary ISA translation[13,32] to address the long tail of a software ecosystem. Such systems are eventually deprecated to avoid carrying forward technical debt, as seen with Apple's planned phaseout of Rosetta.[1] In cloud computing, where compute is commoditized, recompiling onto the new ISA maximizes performance and efficiency. We thus forewent binary translation altogether.

Second, there is extensive research on automatically applying edits to code, such as for performance optimization,[21] fixing security issues,[17] or correcting bugs.[4] As we will see, these are common tasks that are part of a successful ISA migration.

Google's x86 to Arm Journey

We now analyze Google's multiyear effort to port a substantial portion of Google's server application ecosystem from x86 to Arm, enabling simultaneous support for both. We start by describing Google's environment and provide a step-by-step analysis of our ISA migration.

Google's software ecosystem. Google's codebase is organized as a monorepo containing billions of lines of code.[29] Individual applications and libraries reside in various directories. These directories also contain metadata files, for example, to indicate code owners or configure continuous integration (CI) testing.[38] An overview figure of this internal ecosystem can be found in Henderson et al.[15]

Building with Bazel. Builds use Bazel,[6] a highly configurable build system. BUILD and bzl files describe how binaries, libraries, and tests are built from source files. Most code is covered by our primary CI system, "TAP" [Test Automation Platform],[19,24] which gates releases similarly to GitHub Actions or GitLab CI. Bazel's builds and tests (including TAP) run on a shared set of machines called Forge,[37] which provides a distributed build execution and caching service. Forge's scale and cache enable Google to compile everything needed for a binary from scratch on every build, including fundamental dependencies such as the Python interpreter.

Creating releases with Blueprints. Google distinguishes between compiled binaries and release packages (internally referred to as "MPMs" for the Midas Package Manager that stores them globally).[6] These packages are bundles of binaries and static assets ready to be deployed in Google's clusters, akin to Debian packages or container images. Releases are defined by "Blueprint" files, which serve as release configuration manifests (similar to Helm charts or Kubernetes manifests)[25] to specify dependencies, environment settings, and build flags. An automated continuous delivery system called Rapid[5] consumes these manifests, runs CI tests, and builds release packages.

Multiarch support. Google's fleet has supported dozens of CPUs over the years, and services run on machines that can be as much as 10 years old. During development, engineers can configure builds and testing across multiple architectures (e.g., Arm, K8, Haswell) in their Blueprints. Multiple architectures are also supported at release time, producing multiarch MPMs. Unless an owner adds specific constraints, a job can be scheduled on any machine with an architecture-compatible MPM, and owners expect to run on a mix of different kinds of machines with different performance profiles simultaneously.

Shifting down and large-scale changes. Google has moved to a "shift down" approach to development,[11] where developers only focus on one level of the stack and other issues are abstracted and/or automated for them. For example, the CI pipelines (e.g., Bazel, TAP, and Rapid) mean developers do not generally need to worry about the specifics of build systems or releases. In addition, a healthy automated testing culture means everyone can change code throughout the monorepo without frequent breakages. This enables large-scale changes (LSCs),[38] monorepo-wide revisions that can change code owned by many teams over thousands of files at once. Low-risk LSCs can be approved centrally and submitted efficiently without asking individual teams. To manage code edits across many reviewers, Google has developed Rosie,[38] an automated change-orchestration tool that allows engineers to create a very large commit and shard it into tens, hundreds, or thousands of smaller commits split up by the owner.

Life cycle of an ISA migration. Migrating an individual package from

single-arch to multiarch support requires several steps:

1. Test: Fix tests (and builds) that break when run with the new ISA.

2. Set up multiarch CI: Modify the corresponding Blueprint files to ensure no additional regressions are introduced (often simultaneous with the next step).

3. Configure releases: Modify Blueprint files to make releases multiarch by default.

4. Roll out new binaries: Run the multiarch packages on machines of the new ISA and assess performance and stability, addressing issues as needed.

5. Full production: Allow production jobs to be scheduled freely on machines of the new ISA.

While these steps are the same for all packages, the issues encountered within each step vary widely across applications and throughout the different phases of our ISA migration from x86 to Arm. This often involves creating performance-optimized code for the new platform, which can happen in parallel to these steps. We observe parallels to Uber's reported porting workflow.[22]

Phase 1: Large users. We started our Arm migration with a small set of large users, such as the Spanner,[8] BigQuery,[20] and Bigtable[7] big data systems. These migrations were hands-on with a small team and required weekly meetings and tracking bugs. Once tests passed, rollout was manual, with very careful performance and load testing, and gradual scheduling constraint removal on a per-job basis. This first-phase work was done, by and large, by people rather than automation or AI.

During this phase, a number of issues in these workloads surfaced and were addressed. Examples include: x86-specific intrinsics; `long double`, which differs between x86 and Arm, with cross-platform equivalents; brittle tests (e.g., due to exact floating-point equality checks); x86-specific flags; memory ordering issues hidden by x86; out-of-memory errors, often due to heap limits being tuned for x86; multiarch release packages exceeding the capacity limits of our infrastructure; unsupported dependencies, and loading of unsupported dynamic libraries; jobs not getting scheduled due to legacy x86-only scheduling constraints, such as requiring AVX2 instruction support to target newer machines.

Initially, teams perceived porting these mature codebases to Arm as a herculean task fraught with toolchain difficulties. However, as we gained experience, most issues involved simple changes, many of them in configuration files.

This list was surprising to the teams involved. Initially, teams perceived porting these mature codebases to Arm as a herculean task fraught with toolchain difficulties. However, as we gained experience, most issues involved simple changes, many of them in configuration files. At the same time, these changes were surprisingly pervasive, as evident by the 38,156 commits generated.

Generally, we found that the modifications required for software to compile and run on Arm differ from initial expectations. For example, while a package might completely fail to build initially, simple manual fixes to a number of shared dependencies often unblock many builds at once. We also made a few (≈10) systematic, automated changes, such as steering nearly all applications away from `long double` for numeric stability across architectures.

Phase 2: Everybody else. To take full advantage of Arm in datacenters, migrating only the largest workloads is insufficient. To maximize the use of available capacity, the cluster scheduler needs to be able to schedule workloads flexibly across platforms, packing workloads large and small onto machines. If only a small subset of services can run on Arm, it will result in underutilization of those machines. We note that the distribution of workloads at Google is very flat: Although our top 50 binaries are very large, they only represent ≈60% of running compute.[16] Addressing this long tail requires porting more than 100,000 packages and billions of lines of code. This makes the Phase 1 approach of working directly with customer teams infeasible. In fact, even just talking to each team would be prohibitively expensive. The second phase of the x86 to Arm migration therefore focused on automating and scaling the migration of these workloads, while minimizing involvement from the teams themselves. We found that effectively using Arm hardware did not require porting all workloads. It is this phase of Google's x86 to Arm migration that we mostly focus on. So far, we have ported about 60,000 packages, accounting for around 75% of Google's server CPU cycles. In this phase, we

ramped down engineers working on individual tests or configurations, as it was not scalable. Instead, we used a range of human-operated conventional automation tools for a large fraction of commits. Toward the tail end of the process, LLM-generated changes via CogniPort began to emerge, though our analysis in the next section almost entirely focuses on the time before then. We will now describe this phase in more detail, as well as our automation process.

Analyzing an ISA Migration

To fully understand what is involved in an ISA migration, we now analyze the full range of tasks involved in Google's x86 to Arm migration (Research Question 1). As a monorepo, any change—be it to code, configurations, or documentation—is tracked as a commit in our repository's history. Further, relevant commits marked with a keyword indicated they were part of this migration, allowing us to extract them after the fact. We thus identified a relevant set of 38,156 commits. The contents of the changes included code and fixes written almost entirely by humans and conventional (non-AI) automated migration tools, using large-scale changes. The vast majority of the dataset predates the rise of agentic coding tools.

Analyzing these commits manually would have been cost-prohibitive. Instead, we used a variant of Gemini 2.5 Flash[18] (the latest model available at the time) to analyze these commits at scale:

1. We passed the commit messages and code diffs into the LLM's 1M token context window in groups of 100 at a time, and picked 20 categories for each batch.

2. We passed all ≈400 × 20 categories into the context window and asked Gemini to consolidate them into 50.

3. We adjusted the model outputs by hand for a final list of 16 categories (Figure 1).[b]

Code-related changes only account for 1% of commits and less than 4% of lines of code, refuting the conventional wisdom that code translation accounts for most of an ISA migration.

4. We ran Gemini again to apply labels to all of the commits, sorting them into these 16 categories (as well as an additional "Uncategorized" category for outliers, to improve stability). Figure 2 shows examples of each category.

At 38,156 commits, it is impractical to verify that every categorization is correct. We have examined ≈100 exemplars, and they appear correctly categorized.

Commits fall into four overarching groups: code changes, test changes, build and configuration files, and supporting processes and tools. In total, our commits updated around 700K lines of code (LoC). While the vast majority (84%) of commits are related to updating build or configuration files (Category 8), these commits account for only 19% of lines of codes updated. We also see a substantial number of lines (17%) spent on migration tooling. A large portion of this work is only required once and can be reused in future ISA migrations.

Code-related changes (Categories 1–5) only account for 1% of commits and less than 4% of lines of code, refuting the conventional wisdom[26] that code translation accounts for most of an ISA migration. Later in this article, we analyze how automatable these commits are.

We also analyze the timeline of our ISA migration (Figure 3). We observe that at the start of the migration, most commits were in tooling and test adaptation, aligned with Phase 1. Over time, a larger fraction of commits centered around code adaptation, which can be seen as a phase when there is still a need to update code in common dependencies and address common issues in code and tests. Eventually, the fraction of these kinds of commits declined, and in the final phase of the process, almost all commits are configuration files and supporting processes. We also observe that in this later phase, the number of merged commits rapidly increased, capturing the scale-up of the migration.

Why so many configuration changes? Automated and manual flag flips enabled the Arm variant across tens of thousands of projects with simple releases. For projects with complex release testing or evaluation, reconfigur-

b The descriptions in Figure 1 are almost entirely the model's output, with some minimal edits. Note that the model chose to mention alignment; on Google's Arm servers, unaligned access results in a fault because the `SCTLR_ELx.A` bit is set to 1. This typically happened when memory allocated on the stack was declared as an array of `char`, but actually stored a struct.

ing needed more hands-on (or agentic) labor across multiple commits.

Finally, we want to understand how these different categories of commits differ from one another. We observe that median commits in most categories are less than 20 LoC, with many single-line commits. However, we also observe very large individual commits that change 10,000+ LoC and skew the averages. We manually inspected these commits to understand their origin and found these commits do not typically represent more work since they are conceptually similar to large numbers of simple, one-line commits. Overall, there are 19 commits that cumulatively account for 238,289 LoC (32.4%) of total lines changed and that are trivial. Examples include:

- Remove a porting tool once it was no longer used—57K LoC (Category 12)
- Update a list of microbenchmark targets—23K LoC (Category 11)
- Add several very large test vectors for a coverage tool—15K LoC (Category 6)

In summary, we find most commits related to migration are small; larger commits often change very large lists or configurations and are not inherently complex. Finally, we observe that some of the commits (particularly in "Supporting Processes and Tools") could likely be reused in a subsequent multiarch migration.

Automating ISA Migrations

Now that we have established the tasks that are part of an ISA migration, we can explore how automatable each of these tasks is (Research Question 2) and how novel automation approaches can facilitate them.

ISA migration automation at Google. We already employ a number of (non-AI) automation tools at Google today, which automated a large portion of the ISA migration process (83.82% of commits and 14.15% of LoCs).

Figure 1. Categories of commits in Google's x86 to Arm migration. LoC per commit shows median, fifth, and 95th percentile. Automation shows the fraction of commits/LoC generated using large-scale changes.

Change Category and Description	Commits	Total LoC	LoC per Commit	Automation
Code Adaptation and Correction				
1. **Platform-Specific Conditional Code:** Introducing or modifying code blocks that are conditionally compiled or executed (e.g., using #ifdef_aarch64_, runtime CPU feature checks). Examples: Different syscall usage, platform-specific API calls.	236	16,235	9 [1, 330]	22%/28%
2. **Intrinsic and Assembly Code Porting:** Replacing or providing alternative for x86-specific intrinsics (e.g., SSE, AVX) with Arm equivalents (NEON), or rewriting assembly language sections.	74	4,454	10 [1, 209]	1%/0%
3. **Data Representation and Alignment Fixes:** Modifying code to handle issues arising from differences in data type sizes or alignment requirements between x86 and Arm.	38	2,574	9 [1, 324]	0%/0%
4. **Memory Model and Concurrency Adjustments:** Fixing code that makes assumptions about memory ordering, atomicity, or thread synchronization behavior that differ on Arm.	12	49	2 [2, 12]	0%/0%
5. **Performance-Driven Code Optimization for Arm:** Refactoring algorithm or code patterns specifically to improve execution speed, reduce latency, or improve efficiency on Arm microarchitectures. This is beyond basic correctness.	18	2,080	13 [4, 337]	0%/0%
Test Adaptation and Configuration				
6. **Test Logic, Data, and Assertion Modification:** Changing the test code itself, updating golden files, or adjusting test assertions to be compatible with Arm, reflecting valid behavioral differences rather than bugs.	276	37,052	5 [1, 147]	3%/1%
7. **Test Execution Environment and Scope:** Configuring which tests run on Arm, adjusting test timeouts, memory/CPU limits, and sandboxing; excluding tests not suitable for Arm.	1,303	13,783	1 [1, 37]	53%/8%
Build, Deployment, and Infrastructure Configuration				
8. **Build, Packaging, and CI/CD Configuration:** Changes to BUILD files, Bazel settings, genmpm rules, Blueprints, TAP, and release platform configs to support multi-architecture builds, testing, and releases.	32,204	139,611	1 [1, 9]	95%/63%
9. **Borg and Runtime Environment Configuration:** Modifying Borg configurations, allowlists, admission control, resource allocation within jobs, and service enablement for running on Arm in non-prod and production.	757	26,581	10 [1, 136]	1%/0%
10. **Infrastructure Resource Management and Provisioning:** Managing quota, dedicated machines, security policies, storage, network, and scaling for the Arm migration. Includes cluster state management and kernel/platform rollouts.	381	32,159	8 [1, 163]	1%/0%
Supporting Processes and Tools				
11. **Monitoring, Alerting, and Performance Analysis:** Setting up dashboards, alerts, collecting metrics, analyzing performance benchmarks, and classifying errors for the Arm migration.	645	115,343	16 [1, 525]	0%/0%
12. **Migration Tooling and Automation Development:** Creating and enhancing scripts, tools, and automation to assist with any stage of the Arm migration.	644	125,536	29 [1, 808]	2%/1%
13. **Documentation and Knowledge Management:** Creating and updating guides, best practices, and debugging information.	369	24,115	13 [1, 260]	0%/0%
14. **Rollbacks and Cleanup:** Reverting changes and removing obsolete code, configurations, or data from the migration process.	940	163,042	2 [1, 200]	58%/5%
15. **Specialized Service Configuration:** Platform-specific adaptations for databases, experiment frameworks, or other unique services.	119	22,501	7 [1, 747]	3%/0%
16. **Release Qualification and Validation:** Defining and implementing processes to ensure multi-architecture releases meet quality and performance standards on Arm, including CHAMP data.	123	8,178	14 [1, 284]	5%/2%
17. **Uncategorized**	17	1,328	39 [2, 223]	65%/84%

Large-scale changes. A key piece to ISA migration automation is Rosie, a preexisting tool that allows us to programmatically generate large numbers of commits and shepherd them into the monorepo. Rosie orchestrates the testing, mailing, code review, and submission of each individual commit. Rosie managed a total of 31,984 commits, but these changes represented only 14.15% of the total lines changed in our dataset. Figure 1 shows the fraction of each category that was generated using automated tools. For one of the major large-scale automated migrations, we introduced the concept of variant projects to the CI and continuous deployment (CD) infrastructure,[15] which enabled the following single-line addition to configure Arm CI and CD in a Blueprint file:

```
arm_variant_mode =
::blueprint::VariantMode
::VARIANT_MODE_RELEASE,
```

Sanitizers and fuzzers. While not limited to ISA migrations, fuzzers and LLVM sanitizers such as AddressSanitizer,[30] MemorySanitizer,[35] and

Figure 2. Specific code examples for each category.

1. Platform-Specific Conditional Code

```
 #define _CONFIG_H_
 // distro
 #define SOLIB_EXT ".so"
+#if defined(__aarch64__)
+#define ARCH "aarch64"
+#elif defined(__x86_64__)
 #define ARCH "x86_64"
+#else
+#error Unsupported architecture
+#endif
```

Note: Multiarch a config header

2. Intrinsic and Assembly Code Porting

```
+#ifdef __aarch64__
+#include "third_party/sse2neon/sse2neon.h"
+#else
 #include <emmintrin.h>
+#endif
...
+#ifdef __aarch64__
+#define GETROUND()
 ↪  (_MM_GET_ROUNDING_MODE()&VM_SSE_ROUND_MASK)
+#define SETROUND(x) (_mm_setcsr(x|
 ↪  (_MM_GET_ROUNDING_MODE()&~VM_SSE_ROUND_MASK)))
+#else
 #define GETROUND()
  ↪  (_mm_getcsr()&VM_SSE_ROUND_MASK)
 #define SETROUND(x) (_mm_setcsr(x|
  ↪  (_mm_getcsr()&~VM_SSE_ROUND_MASK)))
+#endif
```

Note: Port code using intrinsics

3. Data Representation and Alignment Fixes

```
-char suffix_bytes[k[redacted]FileSuffixSize];
+// NOTE: alignas(uint32_t) is used to ensure
 ↪  proper memory alignment when
+// reinterpreting the char pointer to a uint32_t
 ↪  pointer.
+alignas(uint32_t) char
 ↪  suffix_bytes[k[redacted]FileSuffixSize];
```

Note: Add an alignas modifier

4. Memory Model and Concurrency Adjustments

```
 stripCalcLatch = new CountDownLatch(1);
 calcAwaitingLatch = new CountDownLatch(1);
+ranStripCalc = new AtomicBoolean(false);
 appKey = new PlainKey("fakeKey");
 calculationContext = new CalculationContext(new
  ↪  FakeExtraData<>());
 calcEnv = new InProcessCalcEnv(appKey, ritzInfo,
  ↪  calculationContext);
...
-                       ranStripCalc = true;
+                       ranStripCalc.set(true);
...
```

Note: Switch to atomic boolean to avoid races (Java)

5. Performance-Driven Code Optimization for Arm

```
...
template <>
inline uint64_t MathUtil::SafeRound(float x)
  uint64_t result;
  __asm__("fcvtau %x0, %s1" : "=r"(result) :
   ↪  "w"(x));
  return result;
}

template <>
inline int64_t MathUtil::SafeRound(double x)
  return vcvtad_s64_f64(x);
}
...
```

Note: Implement math routines for aarch64

6. Test Logic, Data, and Assertion Modifications

```
-EXPECT_THAT(distribution.mean(),
 ↪  DoubleEq(event_set.SpaceUsedLong()));
+EXPECT_THAT(distribution.mean(),
 ↪  DoubleNear(event_set.SpaceUsedLong(), 1e-10));
```

Note: Adjust the floating point sensitivity

7. Test Execution Environment and Scope

```
 tags = [
      "cpu:4",
+     "requires-mem:20g",
 ],
 deps = [
```

Note: Increase a test's memory limit

8. Build, Packaging, and CI/CD Configuration

```
 cc_library(
     name = "[redacted]-vector",
     hdrs = [
         "[redacted]-vector.h",
     ],
-    copts = [
-        "-mavx2",
-        "-mbmi2",
-    ],
     visibility = [
```

Note: Remove x86-specific build flags

9. Borg and Runtime Environment Configuration

```
access_level =
-
 ↪  pa_proto.PlatformAllowlistAllowedCollection.
 ↪  INITIAL_LIMITED_QUALIFICATION
+
 ↪  pa_proto.PlatformAllowlistAllowedCollection.
 ↪  EXPANDED_PROTECTION
), {
collection_key = {
user = '[redacted]-ui-mixer'
```

Note: Change allowlist to extend machine exposure

10. Infrastructure Resource Management

```
 borg_pool `[redacted]` = [redacted]arm-dedicated-[redacted]-pool {}
 borg_pool `[redacted]` = [redacted]arm-dedicated-[redacted]-pool {}
 borg_pool `[redacted]` = [redacted]arm-dedicated-[redacted]-pool {}
+borg_pool `[redacted]` = [redacted]arm-dedicated-[redacted]-pool {}
}
```

Note: Add a new cell for testing

11. Monitoring, Alerting, and Performance

```
+perf_[redacted]_workflow_test(
+    name =
 ↪  "borg_workflow_arm_one_build_multi_runs",
+    configuration = {
+        "postprocessing_config": {
+
 ↪  "microbenchmark_[redacted]_data_uploading_config":
 ↪  DEFAULT_MICROBENCHMARK_
 ↪  [redacted]_DATA_UPLOADING_CONFIG,
+        },
+    },
+    # To run workflows on ARM, need to have
 ↪  "--cpu=arm" here.
+    extra_postprocessor_blaze_flags =
 ↪  ["--cpu=arm"],
+    phases = [
...
```

Note: Add a new Arm-specific testing infrastructure

12. Migration Tooling and Automation

```
...
+EXPORTED_ACTIONS = {
+
 ↪  automation_structure_pb2.ACTION_PARSE_BLUEPRINT:
+        parse_blueprint,
+    automation_structure_pb2.ACTION_RUN_TEST:
+        run_test,
+    # Used for testing
+    automation_structure_pb2.
 ↪  ACTION_WRITE_OUTPUT_FROM_INPUT:
+        write_output_from_input,
+    automation_structure_pb2.
 ↪  ACTION_WRITE_OUTPUT_FROM_CONFIG:
+        write_output_from_config,
+}
...
```

Note: Part of a new Blueprint parser

13. Documentation and Knowledge Management

```
...
+    * Borg config managed by the pilot dev team
 ↪  specifically selects the pilot
+        machines for specific pilot jobs.

-* The allowlist of users permitted to run on the
 ↪  pilot machines is managed via
-* the `[redacted]-logs` usermap:
...
```

Note: Evolving the docs for the Logs test

14. Rollbacks and Cleanup

```
 package_name = "[redacted]",
 srcs = [":[redacted]_archive"],
 mpm_tags = ["rapid=[redacted]"],
-platforms = ["//[redacted]/borg:all"],
 deps = [":oneday_[redacted]_borgfiles"],
)
```

Note: Rolls back a problematic data MPM change

15. Specialized Service Configuration

```
+
+// a dedicated borg pool consists of [redacted] and [redacted]
 ↪  machines for A/B testing
+//  - [redacted]: for x86 100%
 ↪  environment
+//  - [redacted]:   for ARM 100%
 ↪  environment
+[redacted]_info [redacted]arm-dedicated-[redacted] =
 ↪  _default_[redacted]_info('[redacted]') {
+  cell = '[redacted]arm-dedicated-[redacted]'
+  user = '[redacted]arm-dedicated'
+  scratch_path_prefix =
 ↪  '/cns/[redacted]arm-dedicated'
+  remote_path = '/cns/[redacted]/home/'
+  platform = '[redacted]'
+}
```

Note: Add a configuration for a Spanner pilot

16. Release Qualification and Validation

```
...
WHEN champ_state = 'Incompatible'
-       THEN PrioritizedStatus(1,
 ↪  'NON_COMPATIBLE_VIA_CHAMP')
+       THEN PrioritizedStatus(0,
 ↪  'NON_COMPATIBLE_VIA_CHAMP')
WHEN champ_state = 'Compatible'
-       THEN PrioritizedStatus(1,
 ↪  'QUALIFIED_VIA_CHAMP')
+       THEN PrioritizedStatus(0,
 ↪  'QUALIFIED_VIA_CHAMP')
...
```

Note: SQL change for improved tracking of qualification

ThreadSanitizer[31] are key enablers of our migration. Even before Arm adoption, Google routinely ran all TAP tests with these tools enabled, which turn latent errors such as memory corruption, memory leaks, or race conditions into debuggable faults. Application owners regularly triage and fix these faults, sidestepping many common differences in execution between x86 and Arm (e.g., a data race may be hidden by x86's TSO memory model). Catching these kinds of issues ahead of time avoids debugging non-deterministic behavior when recompiling to a new ISA.

ThreadSanitizer (TSan), in particular, has proven to be effective in managing the memory model differences between Arm and x86. TSan can detect threading issues in nearly all code except for rare, lock-free algorithms. Programs typically use language-provided synchronization primitives, which fall under TSan's purview and have the same guarantees on both x86 and Arm. Code that has passed TSan is extremely likely to use synchronization primitives correctly.

Although we saw few threading issues across the codebase, some made it through due to code that is not tested or not testable with TSan. Thread-related issues in such code were not easily detectable until we scaled up deployment significantly and saw a pattern of issues, such as crashes. One such issue was a bug in core JVM code[27] caused by a thread race. In our production experience, running at full scale provides high confidence that no issues remain.

Continuous health monitoring platform (CHAMP). The final step in our automation is CHAMP, an automated canary analysis and qualification system that monitors applications running on Arm hardware. It continuously monitors health metrics to detect whether behavior differs from x86 instances of the job (e.g., significantly higher RPC error rates or crashes). If so, it automatically marks the job as ineligible for Arm, files a bug for its owners to follow up, and tries again in 30 days. Following Google's production principles, it scales up the fraction of Arm instances of the application incrementally to limit service-level objective risk. CHAMP is less important for new microarchitecture deployments (either x86 or Arm), as the behavior, performance differences, and associated issues are relatively minor. However, it was highly effective during the Arm migration and helped assure service owners who felt deploying to a new ISA was risky.

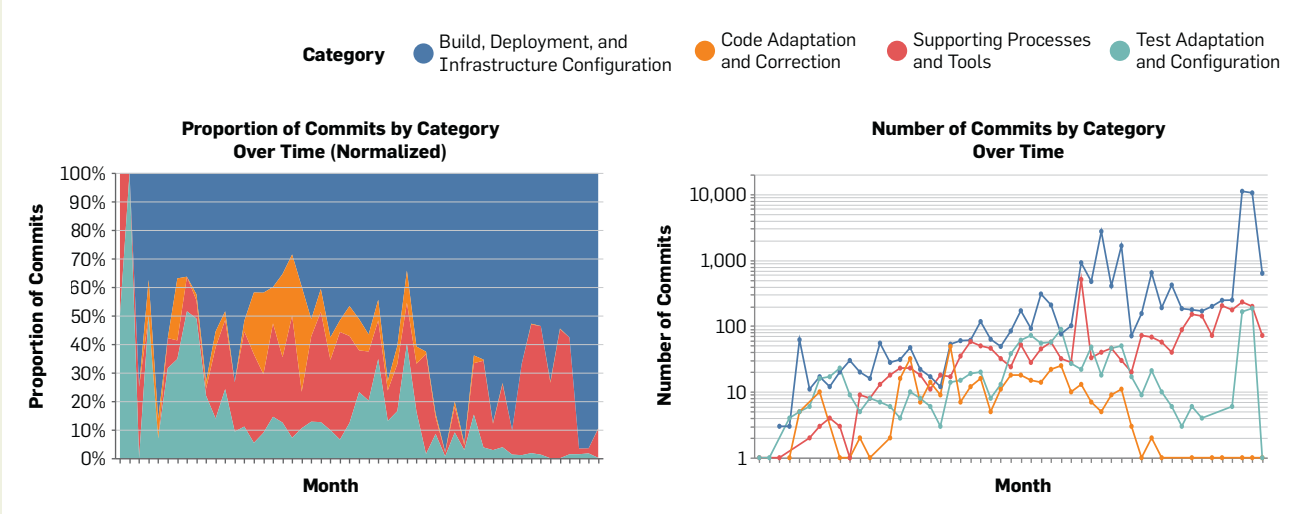
Using CHAMP, it is no longer necessary to manually shepherd every binary through qualification. Instead, after updating project configurations to build Arm releases, rollout is now automatic.

Reliability of the automation approach. Combined, these tools allow for a mostly automated approach where LSCs enable Arm for different builds and releases, which are then automatically qualified using CHAMP. To understand the stability of this approach, we analyze LSCs targeting a particular standardized release management system that covers 49.6% of all binaries at Google. These LSCs modified release configurations to bring this system's percentage of Arm-qualified applications from 4.8% to (at the time of writing) 78.89%. The rate of applications that were rolled back in early testing was 1.8% (which dropped to 1% after fixing bugs), and less than 0.8% in the final phase.

Early in the migration, after ≈300 release packages, we had a 5% refusal rate (code owners deciding not to migrate). During scale-up, this dropped to 0.6% after ≈600 additional packages. In the final phase, the commits were globally approved with no refusal. We found that acceptance rate was strongly influenced by careful workload targeting, users gaining trust in the automation, and messaging that anticipated worries and objections.

The remaining binaries that have not yet been migrated fall into various categories: ready and awaiting migration, uses complex and bespoke testing, has broken tests that have not been repaired yet, on its way to deprecation, uses unmigrated dependencies, is burdened by non-Arm-specific technical debt, has hardware requirements that current Arm servers don't yet fulfill, to name a few. Some small portion of jobs are completely ISA-specific, but the

Figure 3. Categories of commits over time.



vast majority are possible to migrate, and changes to fix them would be classified into our existing 16 commit categories. All of this remains future work for this ongoing project.

Automation of ISA Migrations with AI

While LSCs and CHAMP automate a large part of the process, there are limits to this approach. They can edit build and configuration files, as well as automatically qualify Arm binaries for deployment. However, standard LSCs are fixed-function pipelines. They are not flexible to respond to unexpected errors or other issues that occur at any stage of the process, be it during testing or in production.

Modern generative AI techniques represent an opportunity to automate the remainder of the ISA migration process. We built an agent called *CogniPort*, which aims to close this gap. CogniPort operates on build and test errors. If an Arm binary does not build or a test fails at any point in the process, the agent can step in to fix the problem automatically.

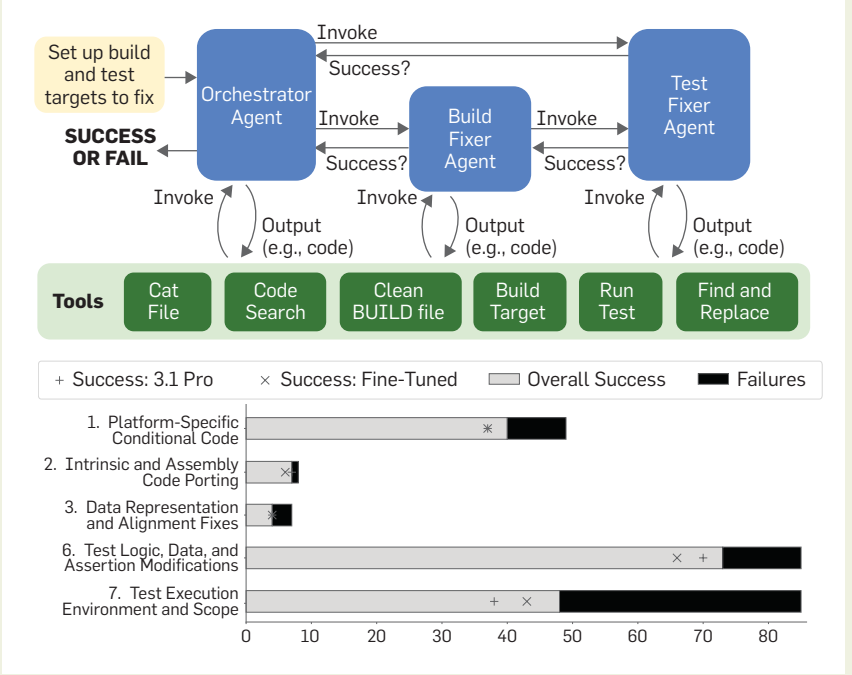
The agent consists of three nested agentic loops (Figure 4). Each loop uses an LLM to perform one step of reasoning followed by a tool invocation—that is, a function call. The tool's outputs are then attached to the agent's context. There are tools for building and returning logs, running test(s) and returning logs, heuristically fixing build file errors, searching through code, modifying code, and exiting the loop (`finish`).

The outermost agent loop is an *orchestrator* that repeatedly calls the *build fixer agent* and/or the *test fixer agent* depending on the state of the workspace. The build fixer agent tries to build a given target and makes modifications to files until the target builds successfully. The test fixer agent runs a given test and makes modifications until the test passes. In both cases, the agent can time out via a step limit or give up early by calling `finish`.

To evaluate the agent, we took historic commits from our dataset, reverted them, and then checked whether the agent fixed the introduced error(s). Note that not all our categories are suitable for this approach—it only applies to Code and Test Adaptation (categories 1–7). We further narrowed down our dataset to commits that can be cleanly reverted and that have identifiable build or test targets. This resulted in a benchmark set of 234 commits in categories 1, 2, 3, 6, and 7. We ran the agent twice for each commit: once with Gemini 3 Flash fine-tuned on a corpus of internal data (including code and documentation)[18,c] and again with out-of-the-box Gemini 3.1 Pro. We saw a best-of-both success rate of 73.5%. Test fixes, platform-specific conditionals, and assembly code porting had the highest success rates. Test Execution Environment was the most difficult but still had a success rate of 56%, with our fine-tuned model performing the best.

Note that we consider these results *directional*: While we analyzed an arbitrary subset of outputs to ensure result validity and confirm the absence of recitation from training, the evaluation is not perfect and may miss cases where, for example, a fix is incorrect but not caught by a test or where there is other information leakage (e.g., a subsequent commit made the original fix easier, or reverting only the commit itself makes it easier to root-cause an issue than if the entire fix was reverted).

Figure 4. Agentic flow and the success rate of the agent.



Discussion and Research Challenges

Overall, we see that ISA migrations involve a wide range of different tasks shown in our taxonomy (Research Question 1). Many of these tasks are highly automatable. Build file and configuration changes are almost fully automatable, while code changes and tests are partially automatable with conventional techniques (Research Question 2). In addition, many areas scale across architectures and do not require additional work for each ISA. This includes multiarch support for build and test infrastructure, spinning up a new hardware platform whether or not it is a new ISA (e.g., categories 11 and 15), and deprecating old code (category 14).

We have released multiarch binaries covering 75% of server compute at Google, a cross-cutting, representative set of code. Historically, such a migration would have consumed immense human effort. Tools like CHAMP and Rosie combined with broad-based testing, liberal use of sanitizers, and

c This corpus may include material related to our Arm migration, but a small enough subset that recitation is not an issue.

focusing on simplifying the process enabled a large fraction of this migration to be automated. CogniPort and similar AI tools will further accelerate the process, making giant codebase migrations tractable at a fraction of the effort. This raises the question: What are the remaining challenges, and are there research opportunities in closing the remaining gaps (Research Question 3)?

By manually inspecting outputs, we find that categories such as build and configuration files, intrinsic and assembly code porting, and test logic are already largely automatable. However, category 7 (Test Execution Environment) proved to be significantly harder. While the details of these challenges are unique to ISA migrations, the root causes are similar to quality issues that arise when using agents to automate general software development. They are rooted in LLM context-window limitations and information-retrieval challenges.

At Google, we found that imperfect AI-based code generation combined with our extensive test and monitoring infrastructure and human code review is sufficient to reliably land large numbers of changes in production, even with a <50% success rate at the model level. For example, a system called ECO[21] has been used to submit thousands of AI-generated code optimizations and less than 0.5% of commits had to be rolled back. The 73.5% CogniPort success rate in our study is already better than rates reported in the ECO paper. Techniques such as information retrieval on past commits (which ECO uses) and spec-driven development (SDD)[12] to make agents more robust could both help further. Even so, we find that a number of open challenges remain:

- *ISA-specific vector code*: Writing ISA-specific, performant vector and matrix code is difficult. Previous research showed limited success.[36] However, we and others[14] observe that modern LLMs are starting to show promise on writing vector intrinsics.
- *Deep performance optimizations*: This sometimes requires major refactors and algorithmic changes. Existing performance-focused AI tools, such as ECO, will also help.
- *Difficult corner cases*: We saw a number of corner cases that require obscure knowledge beyond the code itself. For example, we saw commits that worked around an Arm compiler bug, addressed a hash function behaving differently on Arm and x86, and fixed bugs that were not triggered until the code ran on Arm.
- *Performance tuning*: Hyperparameters and feedback-directed optimization (FDO) profiles sometimes must be regenerated for a new platform. An agent could configure testing harnesses, run workloads, and measure performance.

At Google, we found that imperfect AI-based code generation combined with our extensive test and monitoring infrastructure and human code review is sufficient to reliably land large numbers of changes in production.

Meanwhile, we also found a number of examples not directly related to ISA migration tasks and that may not be completely automatable:

- *Multiarch tooling*: A significant portion of this work includes implementing the automation itself (e.g., CHAMP), as well as simulation tools and dashboards. However, AI agents are rapidly advancing their capability to implement these components.
- *Resource provisioning*: These are configuration changes that help human engineers manage physical hardware in datacenters and can include strategic or business judgement.

Our results demonstrate that the rapid adoption of agentic infrastructure by software engineers, including at Google, will also address ISA migration challenges. Future ISA migrations may require a limited amount of manual work, mostly focused on making new hardware available and adding the new ISA to our existing multiarch tooling and automation.

Conclusion

By analyzing a large-scale ISA migration at Google, we refute longstanding assumptions about such migrations. First, even in the presence of source code, migrations are not straightforward and involve a large range of multifaceted tasks, with no single task dominating. Second, many of these tasks are highly automatable, particularly using modern AI techniques. Finally, AI agents are rapidly improving, expanding the breadth of tasks that can be automated.

Acknowledgments

We thank Arvind Sundararajan, Dushyant Acharya, Ahmed Alansary,

Shelah Ameli, Owen Anderson, Sterling Augustine, Sushmita Azad, Nupur Baghel, Antoine Baudoux, Patrick Bellasi, Vincent Belliard, Kyle Berman, Paul Bethe, Gopu Bhaskar, Raymond 'Princess Sparklefists' Blum, Marshall Bockrath, Harsha Vardhan Bonthalala, Lexi Bromfield, Jean-Luc Brouillet, Eric Burnett, Marcelo Cataldo, John Cater, Kristine Chen, David Cheng, Ilya Cherny, Saahithi Chillara, Rob Chilton, Chandrakanth Chittappa, Brian Chiu, Daniele Codecasa, Eduardo Colaço, Pavithra Dankanikote, Nicolo Davis, Rumeet Dhindsa, Zhuoran Diao, Bartosz Dolecki, Ian Dolzhanskii, Pat Doyle, Elian Dumitru, Ali Esmaeeli, Samuel Foss, Ákos Frohner, Neeharika Gonuguntla, Shruti Gorappa, Russell Gottfried, Manoj Gupta, Benjamin Gwin, Yanyan Han, Jitendra Harlalka, Milad Hashemi, Tim Henderson, Daisy Hollman, Jung Woo Hong, Jiawei Huang, Jin Huang, Talha Imran, Victoria Juan, Pranav Kant, Lera Kharatyan, Joonsung Kim, Danial Klimkin, Justin King, Sree Kodakara, Avi Kondareddy, Danila Kutenin, Gregory Kwok, Pavel Labath, Leon Lee, Sungkwang Lee, Li Li, Sha Li, Xu Li, Zhongqi Li, Jianyi Liang, Kevin Liston, Haiming Liu, Li Liu, David Lo, Sean Luchen, Albert Ma, Laura Macaddino, Anthony Mai, Jennifer Mansur, Simon Marchuk, David Margolin, Alexander Midlash, Dominic Mitchell, Karthik Mohan, Albert Morgese, Maksym Motornyy, Katherine Nadell, Ngan Nguyen, Denis Nikitin, Stoyan Nikolov, Nicolas Noble, Chester Obi, Andri Orap, Habib Pagarkar, Dasha Patroucheva, Sabuj Pattanayek, Vaishakhi Pilankar, Saranyan Vangal Rajagopalan, Daniel Rall, Majid Rasouli, Salonik Resch, Alberto Rojas, Annie Rong, Jesse Rosenstock, Michal Sapinski, Yashnarendra Saraf, Aaron Schooley, Alexander Schrepfer, Manish Shah, Alexander Shaposhnikov, Stan Shebs, Guangyu Shi, Oscar Shi, Santanu Sinha, Anna Sjövall, Ben Smith, Justin Smith, Jairaj Solanke, Fangrui Song, Raman Subramanian, Xenia Tay, Toni Thompson, Dima Tsumarau, Cassie(Yitong) Wang, Tommy Wang, Shu-Chun Weng, DJ Whang, Hailong Xiao, Andy Xu, and Jason Yuan for their contributions to Google's Arm porting efforts and the work described herein.

References

1. About the Rosetta translation environment. *Apple Inc.*; https://tinyurl.com/237khnqb
2. Altman, E.R., Kaeli, D., and Sheffer, Y. Welcome to the opportunities of binary translation. *Computer* 33, 03 (Mar. 2000), 40–45; DOI:10.1109/2.825694
3. Babbage. Apple's Mac Transitions: 68k to PowerPC to Intel to Apple Silicon. The Chip Letter (Substack). (2023); https://tinyurl.com/26sfgmnt
4. Bader, J., Scott, A., Pradel, M., and Chandra, S. Getafix: Learning to fix bugs automatically. In *Proc. ACM Program. Lang.* 3, Article 159 (Oct. 2019), 27; DOI:10.1145/3360585
5. Beyer, B., Jones, C., Petoff, J., and Murphy, N.R. Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, Inc. (2016); https://sre.google/sre-book/table-of-contents/
6. Beyer, B. et al. *The Site Reliability Workbook: Practical Ways to Implement SRE*. O'Reilly Media, Inc. (2018); https://sre.google/workbook/table-of-contents/
7. Chang, F. et al. Bigtable: A distributed storage system for structured data. In *Proceedings of the 7th USENIX Symp. on Operating Systems Design and Implementation*. USENIX Association (2006), 15.
8. Corbett, J.C. et al. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.* 31, 3, Article 8 (Aug. 2013), 22; DOI:10.1145/2491245
9. *Debian Wiki*. RISC-V. (2025); https://wiki.debian.org/RISC-V/
10. Digital Equipment Corporation. *Migrating an Application from OpenVMS VAX to OpenVMS Alpha*. Digital Equipment Corporation, Maynard, MA, (1996); https://tinyurl.com/2c7xdm3z
11. Google Cloud Events. Shift down: A practical guide to platform engineering. YouTube. (2025); https://tinyurl.com/26t4k4nl
12. GitHub. Specification-driven development (SDD). *GitHub*. (2026); https://tinyurl.com/28kzvjc7
13. Gouicem, R. et al. Risotto: A dynamic binary translator for weak memory model architectures. In *Proceedings of the 28th ACM Intern. Conf. on Architectural Support for Programming Languages and Operating Systems 1*. Association for Computing Machinery (2022), 107–122; DOI:10.1145/3567955.3567962
14. Han, L. et al. IntrinTrans: LLM-based intrinsic code translator for RISC-V vector. *arXiv* (2026); https://arxiv.org/abs/2510.10119
15. Henderson, T. et al. Taming the variants multi-architecture continuous testing at Google. In *Proceedings of the 19th IEEE Intern. Conf. on Software Testing, Verification and Validation* (2026); https://tinyurl.com/29z7qw69
16. Kanev, S. et al. Profiling a warehouse-scale computer. In *Proceedings of the 42nd Annual Intern. Symp. on Computer Architecture*. Association for Computing Machinery (2015), 158–169; DOI:10.1145/2749469.2750392
17. Keller, J. and Nowakowski, J. *AI-Powered Patching: The Future of Automated Vulnerability Fixes*. Technical Report. Google LLC (2024); https://research.google/pubs/ai-powered-patching-the-future-of-automated-vulnerability-fixes/
18. Kini, A. et al. Customizing an LLM for enterprise software engineering. *arXiv* (2026); https://arxiv.org/abs/2605.16517
19. Kondareddy, A., Azad, S., Singh, A., and Henderson, T. A. D. Speculative testing at Google with transition prediction. In *Proceedings of the 2025 IEEE Conf. on Software Testing, Verification and Validation*. IEEE (2025), 510–521; DOI:10.1109/ICST62969.2025.10988976
20. Levandoski, J. et al. BigLake: BigQuery's evolution toward a multi-cloud lakehouse. In *Companion of the 2024 Intern. Conf. on Management of Data*, Association for Computing Machinery (2024), 334–346; DOI:10.1145/3626246.3653388
21. Lin, H. et al. ECO: An LLM-driven efficient code optimizer for warehouse scale computers. *arXiv* (2025); https://arxiv.org/abs/2503.15669
22. Lykke, A. and Borlum, J. Adopting Arm at scale: Bootstrapping infrastructure. *Uber Blog* (Feb. 13, 2025); https://tinyurl.com/2ahf9xow
23. Mahon, M.J. et al. Hewlett-Packard precision architecture: The processor. *Hewlett-Packard J.* 37, 8 (1986), 4–21.
24. Memon, A. et al. Taming Google-scale continuous testing. In *Proceedings of the 2017 IEEE/ACM 39th Intern. Conf. on Software Engineering: Software Engineering in Practice Track*. IEEE (2017), 233–242; DOI:10.1109/ICSE-SEIP.2017.16
25. Minna, F., Massacci, F., and Tuma, K. Analyzing and mitigating (with LLMs) the security misconfigurations of Helm Charts from Artifact Hub. *arXiv* (2024); https://arxiv.org/abs/2403.09537
26. MulticoreWare. Why porting applications across architectures isn't simple. (Mar. 14, 2025); https://tinyurl.com/27stoosv
27. OpenJDK. JDK-8369506: Bytecode rewriting causes Java heap corruption on AArch64. OpenJDK Bug Database (2025); https://tinyurl.com/2am2bm26
28. Phillips, P. and Phillips, G. No source code? No problem! What if you have to port a program, but all you have is a binary? *ACM Queue* 1, 6 (Sept. 2003), 50–57; DOI:10.1145/945131.945155
29. Potvin, R. and Levenberg, J. Why Google stores billions of lines of code in a single repository. *Commun. ACM* 59 (2016), 78–87; https://dl.acm.org/doi/10.1145/2854146
30. Serebryany, K., Bruening, D., Potapenko, A., and Vyukov, D. AddressSanitizer: A fast address sanity checker. In *Proceedings of the 2012 USENIX Conf. on Annual Technical Conf*. USENIX Association (2012), 28.
31. Serebryany, K. and Iskhodzhanov, T. ThreadSanitizer: Data race detection in practice. In *Proceedings of the Workshop on Binary Instrumentation and Applications*. Association for Computing Machinery (2009), 62–71; DOI:10.1145/1791194.1791203
32. Shen, B.-Y., Chen, J.-Y., Hsu, W.-C., and Yang, W. LLBT: An LLVM-based static binary translator. In *Proceedings of the 2012 Intern. Conf. on Compilers, Architectures and Synthesis for Embedded Systems*. Association for Computing Machinery (2012), 51–60; DOI:10.1145/2380403.2380419
33. Sites, R.L. et al. Binary translation. *Digital Tech. J.* 4, 4 (Sept. 1992), 137–152.
34. Somogyi, S. and Meng, B. *The PowerPC Macintosh Book: The Inside Story on the New RISC-Based Macintosh*. Addison-Wesley Longman Publishing Co., Inc. (1994).
35. Stepanov, E. and Serebryany, K. MemorySanitizer: Fast detector of uninitialized memory use in C++. In *Proceedings of the 2015 IEEE/ACM Intern. Symp. on Code Generation and Optimization (CGO)*. (2015), 46–55; DOI:10.1109/CGO.2015.7054186
36. Taneja, J. et al. LLM-Vectorizer: LLM-based verified loop vectorizer. In *Proceedings of the 23rd ACM/IEEE Intern. Symp. on Code Generation and Optimization*. Association for Computing Machinery (2025), 137–149; DOI:10.1145/3696443.3708929
37. Wang, K. et al. Scalable build service system with smart scheduling service. In *Proceedings of the 29th ACM SIGSOFT Intern. Symp. on Software Testing and Analysis*. ACM (2020), 452–462; DOI:10.1145/3395363.3397371
38. Winters, T., Manshreck, T., and Wright, H. *Software Engineering at Google: Lessons Learned from Programming Over Time*. O'Reilly Media, Inc. (2020); https://tinyurl.com/25p3pb8d

Eric Christopher was a senior staff software engineer at Google LLC, Mountain View, CA, USA.

Kevin Crossan is a staff software engineer at Google LLC, Mountain View, CA, USA.

Wolff Dobson is a senior staff software engineer at Google LLC, Mountain View, CA, USA.

Chris Kennelly is a distinguished software engineer at Google LLC, Mountain View, CA, USA.

Cannada Lewis is a senior staff software engineer at Google LLC, Mountain View, CA, USA.

Kun Lin is a staff software engineer at Google LLC, Mountain View, CA, USA.

Martin Maas is senior staff research scientist at Google DeepMind, Mountain View, CA, USA.

Parthasarathy Ranganathan is VP, Engineering Fellow at Google LLC, Mountain View, CA, USA.

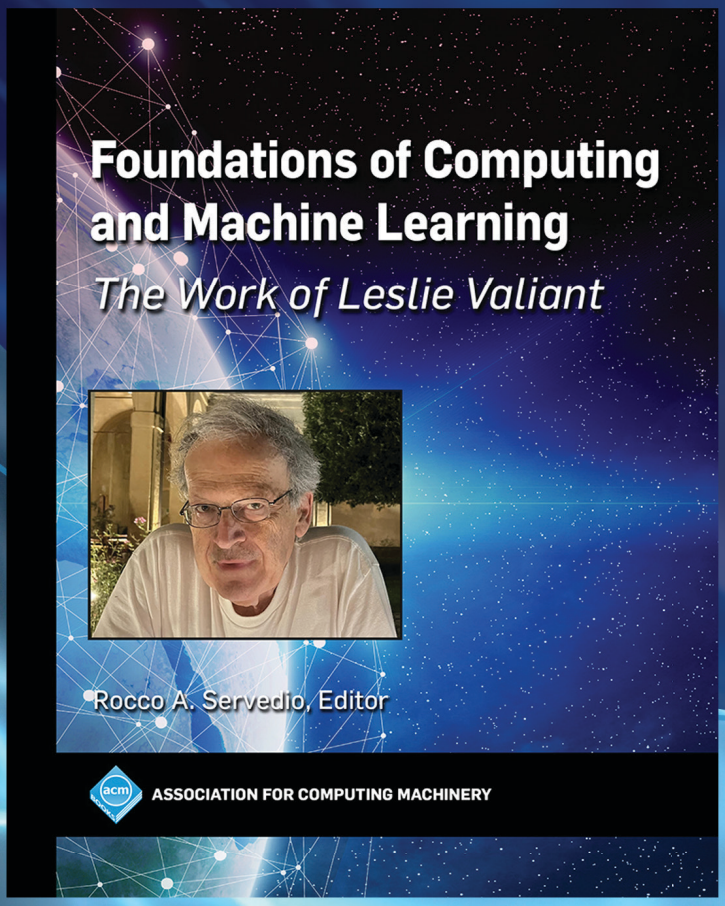
Emma Rapati is a staff software engineer at Google LLC, Mountain View, CA, USA.

Brian Yang is a staff software engineer at Google LLC, Mountain View, CA, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

NEW BOOK RELEASE



ACM BOOKS
Collection III

Foundations of Computing and Machine Learning
The Work of Leslie Valiant

Edited by Rocco Anthony Servedio

ISBN: 979-8-4007-3171-6
DOI: 10.1145/3749740

In 2010, Valiant received the ACM's Turing Award for "transformative contributions to the theory of computation, including the theory of probably approximately correct (PAC) learning, the complexity of enumeration and of algebraic computation, and the theory of parallel and distributed computing."

Leslie Valiant has made deep contributions to an unusually broad range of areas within theoretical computer science. Perhaps his most celebrated contribution has been his paper "A Theory of the Learnable," which gave birth to the field of computational learning theory and provided much of the intellectual foundation for the highly successful development of modern machine learning. His pioneering research in computational complexity inaugurated the study of the complexity of counting problems, defining the fundamental "counting complexity" class #P and showing that a wide range of natural counting problems are complete for this class. Valiant defined algebraic analogues of the complexity classes P and NP and established the first completeness results for these classes; this groundbreaking work opened new vistas for the field of algebraic complexity theory that continue to be explored to this day.

http://books.acm.org

DOI:10.1145/3799433

Though AI offers exciting potential to improve workplace accessibility, it also poses significant risks and can create new barriers for disabled workers.

BY ANNE MARIE PIPER, DARREN GERGLE, CYNTHIA L. BENNETT, MAITREYA SHAH, CHANCEY FLEET, ALI ABDOLRAHMANI, ARIANA ABOULAFIA, STACY M. BRANHAM, SHARON KOPPMAN, ARIELLE SILVERMAN, AND ERIK B. SUDDERTH

AI and the Future of Accessible Work

MANY COUNTRIES HAVE legislation that prohibits discrimination against people with disabilities[a] in the context of employment. For example, the U.S. Americans with Disabilities Act (ADA),[b] the U.K.'s Equality Act of 2010,[c] and European Union policy[d] prohibit employment-related discrimination and require employers to provide reasonable accommodations or adjustments so disabled employees can perform their jobs. More broadly, the United Nations Convention on the Rights of Persons with Disabilities (UNCRPD)[e] has influenced legal frameworks across the globe. Despite these anti-discrimination policies, disabled people remain significantly underrepresented in the workforce. Employment gaps between disabled and nondisabled people exist globally.[28] Among blind and low-vision Americans, for instance, only 44% are employed compared to 79% of those without disabilities.[f] Only 54% of d/Deaf people[g] in the U.S. are employed, compared to 70% of hearing people.[8] These discrepancies are worse for multiply marginalized disabled people. For example, employment rates for disabled women are significantly lower than rates for disabled men.[21]

While many discussions focus on the potential for AI to enhance worker

a There are different norms across disability communities regarding the use of person-first language (e.g., *people with disabilities*) versus identity-first language (e.g., *disabled person*). Knowing that norms and individual preferences vary, we use both person-first and identity-first language throughout.

b https://www.ada.gov/

c https://www.legislation.gov.uk/ukpga/2010/15/contents

d https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32000L0078

e https://social.desa.un.org/issues/disability/crpd/convention-on-the-rights-of-persons-with-disabilities-crpd

f https://www.afb.org/research-and-initiatives/employment/reviewing-disability-employment-research-people-blind-visually

g The use of deaf with a lowercase "d" indicates hearing status while Deaf with a capitalized "D" emphasizes cultural identity. We use the term d/Deaf and hard of hearing to refer to the broader population of people with hearing disabilities.

» key insights

- **A reliance on AI for accessibility may result in increased pressure to use imperfect solutions, leave workers unable to hold companies accountable for accessibility barriers, and obscure the root causes of inaccessibility.**
- **The positive framing of AI for accessibility can mask important issues related to privacy, potential harms, and the need for rigorous evaluation and ethical considerations.**
- **Moving forward requires in-depth research into the real-world use of AI, ensuring a range of accommodation choices beyond AI, and including disabled people in the auditing and development processes.**



productivity and performance, AI can also improve the accessibility of work for disabled people. AI-based innovations enable many assistive applications, including real-time meeting captioning for people who are d/Deaf or hard of hearing, image alt-text generation for blind people (e.g., Microsoft Seeing AI[h]), dictation support for people with speech-language disabilities,[i] and task management for neurodivergent people.[j] AI-enabled tools are also marketed as a way to make digital information accessible, such as via accessibility overlays.[k] Subsequently, AI-based tools are being adopted in workplaces to help address issues of accessibility, with promises of greater autonomy for disabled workers and lower costs of accommodations and compliance for employers.[28]

The exciting potential of AI-powered access technologies, however, may mask essential questions regarding how AI technologies may create additional barriers for disabled people in the workplace and other contexts of use.[33,37] For example, errors and hallucinations in the output of AI models and tools (for a review, see Huang et al.[20]) challenge their use for accessibility in work situations where accuracy is critical.[1,2,36] Digitized hiring assessments, such as AI-scored video interviews, can also be discriminatory toward disabled workers, who may have non-normative speech or body movement patterns, thereby restricting workforce entry.[16,24] Automated tools for accessibility compliance—such as

h https://www.seeingai.com/
i https://www.voiceitt.com/
j https://goblin.tools/

k We note that accessibility overlays are not the preferred or recommended solution for accessibility issues.[26]

ILLUSTRATION BY PETER CROWTHER ASSOCIATES

accessibility overlays—may not work as intended or advertised, making websites more difficult to use and introducing privacy issues.[26]

In this article, we aim to widen the ambit of accessible work by articulating social, organizational, and policy issues that emerge at the nexus of AI, accessibility, and employment for disabled people. We conceptualize accessible work as not only the practical ways in which technology and workplaces are made accessible to people with disabilities but also the ways in which work practices, which are shaped by technology and policy, can both support and restrict disabled people's participation in the workforce. What makes understanding the potential benefits and harms of AI for accessible work challenging is its situated use and embedding within complex organizations with their own evolving norms and policies for both the use of AI and accommodations. Accessibility-related practices and policies are infrequently made public by organizations, and employees may be bound by non-disclosure agreements (NDAs), preventing them from sharing their work-related experiences. Further, employers may require people to disclose their disability to receive workplace accommodations, yet disabled workers may avoid disclosing out of fear associated with stigma or negative repercussions related to job security or promotion opportunities.[34] What's more, disabled people may be required to disclose their disability to AI tools in order to receive useful outputs, such as a blind or autistic person telling an LLM about their disability to get a useful response.[10,22,36] These ongoing challenges become even more complex in light of the increasing rhetoric around AI as a panacea for inaccessible information, technologies, and workplaces.

In what follows, we present a synthesis of the concerns and recommendations raised by researchers, practitioners, and policymakers as part of a one-day workshop on the future of accessible work and AI. Forty-four people participated in the event. Attendees were invited to represent a breadth of expertise and affiliations. Most attendees (n=43) were from U.S.-based organizations, although multiple had experience living and working outside the U.S. Attendees were from academic departments (n=31), including computer science, information science, communication, organization and management, and law schools; industry research labs and product teams (n=7); and nonprofit and government organizations (n=8) that serve people with disabilities. Multiple attendees held appointments across organization types. More than one-third of workshop participants identify as disabled, including people who are blind or low vision, neurodivergent, or who have motor disabilities, chronic illness, or multiple disabilities. Participants also had expertise in accessibility for d/Deaf and hard of hearing people, neurodivergent people (e.g., autistic, ADHD, learning disabilities), and people with cognitive impairment (e.g., dementia).

While discussions considered accessibility and disability in a broad sense, the workshop had a particular focus on the experiences of blind and low-vision people. Workshop presentations and discussions focused on the following motivating questions:

- What is the current state of workplace accessibility given the widespread use of digital productivity and collaboration tools and increasing integration of AI into these tools?
- What are key technical challenges and emerging approaches to AI that have the potential to transform accessibility and the future of work?
- What organizational, policy, legal, and ethical issues are central to creating an accessible future of work as AI continues to evolve?

The resulting themes below were derived by analyzing meeting transcripts and notes, presenter slides, and follow-on conversations with workshop participants who are experts in accessibility, AI and machine learning, organization science and work studies, and disability and AI policy. Our analysis focused on identifying cross-cutting themes that emerged during discussion, grounding them in concrete examples offered by participants regarding their lived experiences with issues of AI and accessibility in the workplace. Workshop attendees were made aware that the meeting discussions were being transcribed and would be analyzed as part of preparing a public report on the workshop's findings. This project was deemed "non-human subjects research" by the lead author's university IRB. To avoid identifying individual participants, we do not provide detailed demographic data and present our findings in aggregate below. Given the limitations of our sample of participants, we contextualize the consequences and recommendations discussed below by presenting them alongside related academic research and policy efforts and enrich them with first-person accounts of how disabled workers experience these challenges in practice.

Consequences of AI for Accessible Work

The consequences of AI for accessible work are not merely technical, such as model biases, errors, and hallucinations in output that can be misleading and problematic when used in the workplace. They are also structural, including the potential for increased and systemic discrimination of disabled workers; organizational, through evolving workplace norms and policies on the use of AI and accommodations; and social and emotional, through how AI-powered technologies have the potential to affect experiences of workplace acceptance and belonging. Below we articulate six consequences of AI for accessible work.

Workers are likely to feel pressure to use AI-based accommodations. Disability policy, particularly in the U.S., U.K., and EU, provides disabled people not only the right to reasonable accommodations but also the right to refuse an accommodation if they so choose. Yet we know that disabled people already report hesitancy in telling their employers if or when they need accommodations, for fear of backlash or retribution.[34] Multiple participants explained that, with the promises of AI for accessible work, there is likely to be increased social and organizational pressure on people to use AI-based technologies to meet their access needs, even if the technologies are less ideal than human support. For example, one blind participant explained that blind and low-vision people can feel pressure to use an AI image-description tool, such

as Seeing AI, rather than ask a human for a visual description, even if the AI-based tool is imperfect and contains inaccuracies. Participants raised the parallel issue of d/Deaf and hard of hearing people feeling pressured to use relatively inexpensive automatic captioning in place of more accessible, yet more expensive, human sign language interpretation or human-generated communication access real-time translation (CART) captions. This increased pressure is likely to stem from employers wanting to cut costs, and from employers and disabled workers having different perspectives on what constitutes "reasonable" accommodations, particularly as AI-based accommodations become widely used in certain work contexts. Moreover, participants cautioned that when employees feel pressured to adopt AI-powered accommodations rather than the tried-and-true solutions they find most effective for given work tasks (such as human assistance), AI may ultimately become the de facto accommodation. Participants called for accommodation processes to be participatory, individualized, and determined according to what will foster an accessible workplace.

Disabled workers may be unable to hold employers accountable for inaccessible AI-based applications. While current policies in multiple countries provide legal mechanisms for workers to hold employers accountable for accessibility in the workplace, enforcement of that accountability is not straightforward. Multiple participants emphasized that even if a worker feels comfortable pointing out accessibility issues in their workplace, some countries have loopholes or exceptions that allow businesses to skirt responsibility. Referencing U.S. policy, one person explained that if a blind worker cannot access a timekeeping application, but the timekeeping application is a third-party software created by another company, the worker has no legal recourse for ensuring accessibility through their employer.[l] Other loopholes exist for small businesses or when accessibility compliance would place an undue financial burden on the business. For instance, in the EU Employment Directive, there exists a "disproportionate burden" exception to the mandatory reasonable accommodations provision[m] that allows for cost considerations in employer decisions.

With the promises of AI for accessible work, there is likely to be increased social and organizational pressure on people to use AI-based technologies to meet their access needs, even if the technologies are less ideal than human support.

Participants, primarily those from industry and with legal/policy expertise, raised concerns that many software products and services that integrate AI exist within those loopholes, making it difficult for workers to hold employers, or anyone else, accountable when an AI tool is inaccessible or creates accessibility problems. The challenge is not only that workers may have no legal recourse, but also that the companies disabled people work for may need to spend time tracking down where the accessibility issue originated—and who is responsible for fixing it—before remediation can happen. This problem gets exacerbated in countries like the U.K., where the law does not prescribe a definite timeframe within which employers must make reasonable adjustments for employees.[n] Several participants emphasized that the closed-box designs of many AI systems and stochastic nature of generative models make it difficult for even the manufacturers of AI products to identify what is creating the access barriers, let alone companies or individual employees working with third parties to try to diagnose the issue.[30] Collectively, participants expressed concern that AI employed as a workplace accommodation has the potential to rapidly scale these issues, leaving disabled workers unable to hold employers accountable and enabling businesses to avoid direct responsibility for creating inclusive environments.

AI promises to resolve accessibility issues without addressing the root causes. One key promise of AI for accessibility is to remediate—or resolve—accessibility problems in a variety of situations, from inaccessible

l In the U.S., this loophole is closed by the ADA Title II for state and local government entities, although many employers do not fall under that criteria.

m See Recital 21 of the disability nondiscrimination directive: https://eur-lex.europa.eu/eli/dir/2000/78/oj

n https://www.edf-feph.org/content/uploads/2024/10/united_kingdom-accessible_DATA_project.pdf

meeting environments to websites and software. For example, many accessibility overlays use AI to re-render digital content in an attempt to make websites and applications more accessible for disabled users.[o] Related approaches improve mobile application accessibility by generating missing metadata based on visual screenshots,[40] a feature that was added to Apple iOS VoiceOver.[p] Despite the potential of AI to remediate inaccessible digital experiences, multiple participants highlighted that accessibility overlay companies have overpromised the potential of this approach to resolve accessibility issues, resulting in litigation and significant fines (e.g., AccessiBe[q]) Even worse, accessibility overlays may increase challenges for some assistive technology users by blocking keyboard input or failing to accurately detect document structure.[26,31]

Multiple discussions emphasized that turning to AI to resolve accessibility issues also has the potential to leave the root cause of inaccessibility unaddressed and create other problems. Accessibility overlays, for example, can threaten user privacy, misapply semantic structure, mislabel elements, and fail to detect nuance in graphical information.[26,31] Workshop participants raised concerns around how disabled users will verify real-time remediation, such as a blind person needing to understand whether the presented overlay is the correct equivalent of the visual layout. While AI-based techniques for remediation will continue to improve, turning to AI as a post hoc fix does little to ensure that software and Web content abide by the Web Content Accessibility Guidelines (WCAG)[r] from the start. Instead, participants explained that AI-based remediation may create the perception that AI will be there to "clean up" inaccessible code, reinforcing a mindset that coding with accessibility standards in mind is optional and reducing market pressure for companies to make technologies accessible from the beginning.

When inaccessible code is used as training data, it allows inaccessibility to propagate through and be reinforced by AI-based coding tools.

Participants suggested that the perception that AI will be there to fix and clean up inaccessible technology has the potential to derail more than two decades of social, educational, and technical efforts to build accessibility awareness among different stakeholders. Natively accessible software and Web content is not only important for equal access and ensuring a consistent user experience for all people, but is also tomorrow's training data. One participant pointed out that when inaccessible code is used as training data, it allows inaccessibility to propagate through and be reinforced by AI-based coding tools. A recent experiment that used an LLM to generate website code similarly showed that 84% of generated websites contained accessibility violations.[3] What's more, novice developers using AI for programming support may fail to prompt for help with accessibility or miss crucial steps required for making code accessible.[27] As AI becomes normalized as a way of resolving accessibility issues in code, participants emphasized the importance of considering the consequences of this mindset on the entire ecosystem of development tools, coding practices, and software review guidelines.

A reliance on AI may undermine the collective effort required for access. Accessible work requires more than simply ensuring digital experiences pass certain accessibility benchmarks (e.g., WCAG compliance). Creating accessible workplaces and work practices involves ongoing awareness and understanding of access needs among one's colleagues and organization.[6] Similar to developers relying on AI to clean up inaccessible code, workshop discussions emphasized that an over-reliance on AI for making work accessible has the potential to undermine the human efforts and mutual understanding that are crucial for effective collaboration among ability-diverse teams in the workplace (see also Branham and Kane,[9] Cha et al.,[11] and Das et al.[12]). Multiple workshop participants noted that with greater reliance on AI for making work accessible, coworkers may perceive AI as meeting a disabled coworker's access needs and feeling

o https://karlgroves.com/overlay-fact-sheet-what-it-is-and-how-to-make-it-work-for-you/
p https://support.apple.com/en-us/111799
q https://www.lflegal.com/2025/01/ftc-accessibe-million-dollar-fine/
r https://www.w3.org/WAI/standards-guidelines/wcag/

less need to adapt their own behavior to support workplace access. Thus, accessibility is further pushed onto the disabled worker as an individual burden rather than a collective effort.[6] One example discussed by participants involves using AI to make virtual meetings more accessible by providing captioning and auto-generating meeting notes. While this may, in fact, be helpful to neurodivergent, d/Deaf and hard of hearing, and blind or low-vision workers, meeting accessibility involves more than technology. All meeting attendees must be aware of and uphold accessible practices, such as being mindful of turn-taking when speaking and distributing meeting agendas and materials in advance to allow time for review.[14] Workshop participants emphasized that while AI may ultimately shift some of the labor of access from humans to machines, we have yet to understand the consequences of that shift in terms of who is viewed as responsible for creating access—whether it is AI, employers, coworkers, or disabled employees themselves. Multiple participants described how disabled people are often expected to "bring their own accessibility" or individually take on the work of figuring out how to ensure their access needs are met. Thus, the normalization of AI as a one-stop shop for accessibility may ultimately put more social pressure on disabled people to sort out their own access needs rather than shifting the pendulum toward access as a collective phenomenon.[6]

Disabled workers may be unable to opt in or opt out of using AI. Workshop discussions highlighted the ways in which AI-augmented workplace tools promise to support new ways to generate and refine content, derive insights from data, and engage with customers, ultimately boosting worker productivity and performance. As such, participants shared stories of organizations rapidly adopting new AI-augmented workplace tools, often ushering in new tools without fully checking their accessibility compliance. One participant shared their experience as a blind U.S.-based employee joining a new team at work. The team uses an AI-augmented digital whiteboard tool, such as Miro, to coordinate tasks and track project deliverables, yet this tool is not fully accessible to them. Not only is it tedious to add and modify content with their screen reader, they also face difficulty maintaining both local and global understandings of the collaborative workspace given its highly visual nature. When monitoring a team's status and coordinating tasks depends on understanding inaccessible visual representations, they are unable to effectively do their job. Rather than the AI-enabled tool bolstering productivity, it may instead directly hinder their ability to participate, with consequences for their actual and perceived job performance.[11] Moreover, the employee is faced with the impossible situation of either asking their team to change their tool selection to accommodate their needs, with the potential for backlash and being viewed as not a "team player,"[12] or to shoulder the burden of finding workarounds that may still limit their participation.

As AI becomes normalized as a way of expediting work and offloading certain tasks, participants explained that disabled workers may feel increasing pressure to use AI just to keep up. One attendee shared the example of a worker, when feeling particularly fatigued or overwhelmed, deciding to offload a task to AI, such as gathering information across a range of websites rather than spending hours sifting through partially accessible content. They explained that the better these tools get, the easier and more tempting that decision is for all people—but especially for people with disabilities, who are dealing with elevated levels of friction every single day because of inaccessibility. To further complicate matters, multiple participants emphasized that disabled workers may fear becoming overreliant on AI to support their work and then having their access revoked. Certain companies or roles may not allow its use, such as when working on a proprietary and confidential code base, or may change their policies around the use of AI tools and suddenly prohibit the use of a tool that has become integral to one's workflow.[10,11] Workplaces are rapidly adopting new AI applications, and in parallel evolving policies for their use, creating ongoing tensions that may result in disabled workers being unable to opt in or out of using AI for access.

The beneficence of AI for accessibility may elide possible future harms. Throughout workshop discussions, participants highlighted the potential for AI to make a positive social impact by increasing access to information and giving disabled people more choices for meeting their access needs. The beneficence of AI for accessibility is a powerful message and marketing tool, explained one participant, positioning subsequent advances as not only a necessity but also a human right. Participants warned, however, that this perspective can obscure or elide concerns over the implementation and use of AI tools, such as how data protections and privacy are managed, as well as how even accessible AI tools can discriminate against people with disabilities in other ways. The trade-off between privacy and access is not new to disabled people, as they are routinely forced to give up their privacy to secure access or give up access to maintain their privacy.[35] A poignant question one participant posed to the group is whether accessibility as a use case for AI nudges people toward accepting a technology that until now would have felt highly invasive.

Another key concern raised during discussion involves whether the normalization of AI among youth means we are socializing the next generation of workers to accept such technologies in contexts where similarly invasive technologies would be unwelcome. For example, if AI tools are socialized among youth as appropriate within classrooms—a sensitive context of use—future generations of workers may come to accept such technologies in equally sensitive contexts within the workplace, such as private meetings and discussions. Multiple participants emphasized that we must be mindful of how the promise of AI to improve accessibility, whether in workplaces or classrooms, has the potential to normalize surveillance culture and obfuscate potential harms.

Another consequence of the beneficence of AI for accessibility is that certain requirements and review processes may be relaxed or fast-tracked when motivated by narratives of social good, with participants sharing examples of

this already happening in their organizations. That is, participants noted that accessibility as a use case may be leveraged to skirt technology reviews or lessen requirements around data retention and privacy. Several participants expressed that within an employment context, the risks are exacerbated for disabled workers, who may need to use fast-tracked, minimally tested AI tools to meet their access needs. Multiple participants with legal expertise noted that these same tools may reveal a worker's disability status to their employer, among many other potential harms.[38] Data that captures the way people walk, talk, look, and interact can all be used as proxy data to infer disability status, potentially exposing protected disability information in an employment context.

Toward Responsible AI for Accessibility in the Workplace

How AI will shape accessible work for disabled people—for better or worse—presents more questions than answers at this stage. The use cases, norms, and concerns of AI for the future of work are rapidly evolving, creating new considerations for disabled workers, who may simultaneously depend on and deplore AI tools. Policies and regulations on both AI and disability vary locally and globally, are being actively negotiated, and are often in tension, yielding continually shifting currents that disabled workers must navigate. Grounded in our findings and related literature, we make four recommendations for moving toward responsible AI for accessible work.

Understand the situated uses of AI in the workplace among disabled people. Studying the ways in which AI technologies are adapted and appropriated within a wide range of workplaces and settings is essential to understanding their immediate and longer-term impacts on disabled workers' ability to secure and maintain employment. Use of AI technologies must be contextualized within organizational settings to understand how it may both alleviate and create new tensions for disabled workers. It is well known both that disabled people are early adopters of technologies and that newly introduced technologies often have unanticipated negative effects on workers. For instance, organizations have implemented algorithmic systems in the workplace to improve coordination and decision making, yet these algorithms have similarly intensified work and managerial control, leading to negative consequences such as increased stress, job precarity, and loss of privacy.[23] It is also important to go behind the scenes and actually see how AI is being used in practice. New technologies are often valued more for what they symbolize than how they are actually used.[4] As a result, there is often a gap between the two. For example, one study found that hospitals promoting their use of surgical robots in marketing campaigns actually kept these robots locked away in storage closets,[5] while another found that startups often exaggerated the effectiveness of new software algorithms to secure funding.[32] Given the widespread hype surrounding AI, and AI for accessibility in particular, there is likely a substantial gap between what firms say AI-enabled tools can accomplish and how they are used on the ground. The fines against AccessiBe for overpromising their ability to make websites accessible may be the first of many examples.[q]

Prior work shows there are certain situations where disabled people prefer and disprefer using AI as an accessibility tool (e.g., Adnin and Das,[1] Carik et al.[10]), yet what is missing is an understanding of how workplace expectations and norms, which vary across countries and cultures, may increase the pressure to use AI and its associated risks. Working in collaboration with disabled people to study the situated uses of AI for accessibility in the workplace also has the potential to identify practical evaluation metrics or benchmarks. For example, even if AI performs well on general benchmarks evaluating its average performance, it will be a poor-quality accommodation if the model or resulting applications contain biases around disability or cannot remediate certain content. Benchmarks must therefore be contextualized and applied according to spectra,[17] considering factors such as task importance, the tool's utility to provide reliable information, and whether it is used alongside other accommodations (e.g., human verification). Situated studies of AI for accessibility will not only help identify meaningful benchmarks but also reveal potential harms and shortcomings, such as how errors or model hallucinations may affect job performance, or the follow-on human labor required when AI tools only partially remediate inaccessible content. As AI is assessed for its effectiveness and suitability as an accessibility accommodation, it is essential to directly involve a wide variety disabled people, including multiply marginalized disabled people, as primary stakeholders. Their perspectives are crucial to understanding how well AI tools function in situ and to identifying potential harms, ranging from personal and data privacy concerns to impacts on worker self-efficacy, across diverse workplace contexts.

AI-powered accommodations should not be the only option. While disabled workers in many countries have the right to reasonable accommodations in the workplace, the proliferation of AI-based accessibility tools is likely to complicate what is considered "reasonable" by various stakeholders. AI-based accommodations can be empowering to people with disabilities by giving them more agency and privacy to do their work, and they can be more cost-effective for workplaces needing to provide accommodations. Yet many AI-based access technologies exist to remediate content that was not born accessible, thus serving as an imperfect ad-hoc solution that, in many cases, is better than nothing but still lacking. As AI becomes an accepted accommodation option, we must be mindful of how AI will be perceived as a "reasonable accommodation" by various stakeholders in certain contexts and as wholly inadequate and inappropriate in others. When accommodations are inadequate, employees are often left with the only option of filing an individual lawsuit against an employer, which is time-consuming, expensive, and puts the burden of proof on the employee. Policies and monitoring mechanisms should proactively consider the implications of AI on accommodation processes.

Moreover, accommodations are not one size fits all, but legally and prag-

matically should consist of multiple solutions that employees may leverage depending on the task demands. For example, a blind employee may prefer to use an AI-enabled tool to draft content and choose a first round of images to put into a presentation, but then consult a human visual interpreter when refining the presentation. Neurodivergent workers may prefer AI for some social communication tasks and not others.[10] As AI becomes more normalized as an accommodation, it is critical that workers maintain the ability to fluidly move between accommodation options depending on the task, context, and what is at stake. Though AI can be a valuable accommodation option, it should not be the only option.

There is a growing need for a range of both human and automated accommodation options, especially in situations where users cannot independently verify a tool's output. While disabled people often rely on human assistance to confirm critical information produced by AI tools,[1,2,36] an increasing reliance on AI raises concerns about how scalable and sustainable this verification process is. Contemporary LLMs and VLMs remain prone to errors, and much of their practical success in application stems from their ability to quickly and easily generate "initial" content that can be verified and refined or edited by a human. Given the high cost of human verification, one approach could be for AI tools to signal uncertainty, although even the most advanced AI models today struggle to reliably assess how confident they are in their own outputs.[39] Further complicating verification, many scenarios of use have no single "correct" answer, such as an autistic worker needing to evaluate the quality of job-related advice from an LLM.[22] Across the broader spectrum of disability, we know even less about how workers with intellectual disabilities (e.g., Down syndrome) or mild cognitive impairments (e.g., early dementia) understand and ensure the quality of output, particularly when recommendations from AI and human experts diverge.[22]

Involve disabled people in efforts to audit AI technology. Early work on accessibility and AI called for greater involvement of disabled people in the development of AI systems, including sourcing data to build models and surfacing issues of fairness.[37] Related efforts infuse disability perspectives into AI development pipelines with disability data, for example, by releasing disability datasets[18] and proposing infrastructures for accessible data collection.[29] These efforts, while a valuable step forward, do not fully engage disabled people in efforts to audit AI technology. Participatory AI (PAI) is gaining momentum in HCI and AI communities.[19] A component of PAI, community-based auditing, empowers the people most impacted by AI to lead in evaluating it. However, accessibility research has long pointed out discrepancies between calls for participatory design processes and their success when applied in disability communities. For example, the methods and tools used in these processes may themselves be inaccessible (e.g., sticky notes,[7] digital whiteboards[13]). A first step in community-based auditing of AI in the workplace is to ensure that the auditing process and training in foundational machine learning (ML) and auditing concepts are accessible. For example, researchers who evaluated AI-generated images depicting disability provided image descriptions to blind evaluators;[25] one challenge in making this material accessible concerns when people do not have access to AI outputs without them being translated, and when AI itself is serving as an accessible remediation of inaccessible "ground truth." Additionally, if workplace technologies are being audited, extra care is required to ensure employees are not obligated to conduct these audits in ways that protect corporate interests, and that they are free to audit the AI honestly. Engaging third parties to facilitate audits may be necessary. Companies could alternatively sponsor community organizations to lead audits; this could be an opportunity for multiple organizations using similar AI capabilities to benefit from a public audit. Disabled workers should also be empowered to audit AI-enabled tools not just for accessibility, but also for other barriers that may lead or contribute to disability discrimination in the workplace.

It is critical that workers maintain the ability to fluidly move between accommodation options depending on the task, context, and what is at stake. Though AI can be a valuable accommodation option, it should not be the only option.

Finally, static and discrete notions of disability challenge the very concept of auditing as it is commonly configured in ML research. As disability is dynamic and every person's experience of it is affected by their unique position, and since people's diverse accessibility needs require information to be presented in multiple formats, audits will need to be flexible to capture evaluations by disabled people, as compared to benchmarks with only predetermined metrics with closed-ended response options.[15,25] What's more, we are all enmeshed in a global work product, which is the unseen labor of tuners, moderators, data refiners, and others. We know little about who is doing this work and whether they are fully representing the disability community. The people making decisions at this level may not be people with lived disability experience, and if they are, they are bringing their own disability culture to bear on decisions. Moving forward, another open question is how well existing accessibility audit tools,[s] which are useful for assessing many types of content and experiences for a broad range of users, extend to and align with emerging AI auditing processes and frameworks.

Disability must be a key consideration in AI governance and policy. AI can provide unique benefits to people with disabilities, and these benefits should be considered whenever policies are developed, particularly when they limit disabled people's use of AI. For example, policies within organizations may restrict employees from using certain AI products, even if those same products provide useful accommodations for a disabled employee. While AI should not be the only option for accommodations, its use should also not be limited when there is benefit as an accommodation, particularly without a compelling reason (like worker safety or security). In contrast, if there are certain cases where AI is limited to accessibility use cases, policy must also outline how someone justifies usage and whether that requires disclosing disability, which presents additional barriers. Conversely, accessibility should not become a loophole for inappropriate usages of AI, such as by companies fast-tracking use cases of AI for accessibility and circumventing rigorous review.

One way forward is to take a dual track approach, in which disability is included in mainstream AI governance policy and AI is included in core disability policy.

One way forward is to take a dual track approach, in which disability is included in mainstream AI governance policy and AI is included in core disability policy. The EU AI Act makes progress toward AI governance and policy that attends to disability.[t] It prohibits developing systems that exploit disabled people, emphasizes making systems accessible for disabled users, and encourages developing codes of conduct that assess and prevent the negative impact of AI systems on accessibility for disabled people. As another example, in the U.S., the New York Artificial Intelligence Consumer Protection Act proposes third-party auditing of AI-based hiring tools with attention to discrimination based on gender, race, disability, and other protected classes.[u] While these policies provide an important step forward, working toward meaningful AI governance and policy that includes disability is challenging, due to limited population-level data on disability and the underrepresentation of disabled people and associated data in AI models.

Conclusion

The consequences of AI for accessible work extend beyond mere technical considerations, shaping and being shaped by organizational norms and impacting the social-emotional experiences of disabled workers. Throughout this article, we highlight not only how technology and workplaces are made practically inaccessible, but also how work practices—influenced by AI adoption and policy—can both enable and constrain disabled people's participation in the workforce. The pressure to adopt AI-based accommodations, the challenges of accountability for inaccessible AI, the erosion of collective responsibility for access, and the obfuscation of potential harms all necessitate a cautious and critical approach. While we surface many ongoing and future chal-

s https://www.w3.org/WAI/test-evaluate/tools/list/

t https://artificialintelligenceact.eu/

u https://www.nysenate.gov/legislation/bills/2025/A768

lenges of AI for accessible work, more research is needed to fully understand how such issues manifest for various groups of workers across the globe. Our sample is limited in that participants were primarily U.S.-based, and those coming from industry are from larger technology firms or nonprofit organizations. Additionally, blind and low-vision people were more heavily represented than other disability groups. Moving forward, more research is needed to ensure that a diverse range of disabled workers from across the globe are actively involved in surfacing issues regarding contextualized use of AI for work, the auditing of AI technologies, determining evaluation metrics and benchmarks, and developing robust governance and policy frameworks. Only through such concerted and holistic efforts can we hope to harness the transformative potential of AI in a way that genuinely empowers disabled people and fosters inclusive and equitable workplaces. C

References

1. Adnin, R. and Das, M. "I look at it as the king of knowledge": How blind people use and understand generative AI tools. In *Proceedings of the 26th Intern. ACM SIGACCESS Conf. on Computers and Accessibility*. ACM (2024), 1–14.
2. Alharbi, R. et al. Misfitting with AI: How blind people verify and contest AI errors. In *Proceedings of the 26th Intern. ACM SIGACCESS Conf. on Computers and Accessibility*. ACM (2024), 1–17.
3. Aljedaani, W. et al. Does ChatGPT generate accessible code? Investigating accessibility challenges in LLM-generated source code. In *Proceedings of the 21st Intern. Web for All Conf.* (2024), 165–176.
4. Anthony, C., Bechky, B.A., and Fayard, A.L. "Collaborating" with AI: Taking a system view to explore the future of work. *Organization Science* 34, 5 (2023), 1672–1694.
5. Beane, M.I. In storage, yet on display: An empirical investigation of robots' value as social signals. In *Proceedings of the 2020 ACM/IEEE Intern. Conf. on Human-Robot Interaction*. ACM (2020), 83–91.
6. Bennett, C.L., Rosner, D.K., and Taylor, A.S. The care work of access. In *Proceedings of the 2020 CHI Conf. on Human Factors in Computing Systems*. ACM (2020), 1–15.
7. Bennett, C.L. et al. Using a design workshop to explore accessible ideation. In *Proceedings of the 18th Intern. ACM SIGACCESS Conf. on Computers and Accessibility*. ACM (2016), 303–304.
8. Bloom, C.L., Palmer, J.L., and Winninghoff, J. Deaf postsecondary data from the American Community Survey. University of Texas at Austin (2023); https://nationaldeafcenter.org/news-items/supporting-deaf-people-closing-the-employment-gap/
9. Branham, S.M. and Kane, S.K. The invisible work of accessibility: How blind employees manage accessibility in mixed-ability workplaces. In *Proceedings of the 17th Intern. ACM SIGACCESS Conf. on Computers & Accessibility*. ACM (2015), 163–171.
10. Carik, B., Ping, K., Xing, X., and Rho, E.H. Exploring large language models through a neurodivergent lens: Use, challenges, community-driven workarounds, and concerns. *Proc. ACM Hum.-Comput. Interact.* 9, 1 (2025).
11. Cha, Y. et al. "Do you want me to participate or not?": Investigating the accessibility of software development meetings for blind and low vision professionals. In *Proceedings of the 2024 CHI Conf. on Human Factors in Computing Systems*. ACM (2024), 1–17.
12. Das, M., Gergle, D., and Piper, A.M. "It doesn't win you friends": Understanding accessibility in collaborative writing for people with vision impairments. *Proc. ACM Hum.-Comput. Interact.* 3, CSCW (Nov. 2019), 191:1–191:26.
13. Das, M., Stangl, A., and Findlater, L. "That comes with a huge career cost:" Understanding collaborative ideation experiences of disabled professionals. *Proc. ACM Hum.-Comput. Interact.* 8, CSCW1 (Apr. 2024), 179:1–179:28.
14. Das, M., Tang, J., Ringland, K.E., and Piper, A.M. Towards accessible remote work: Understanding work-from-home practices of neurodivergent professionals. *Proc. ACM Hum.-Comput. Interact. 5*, CSCW1 (Apr. 2021), 183:1–183:30.
15. Gadiraju, V. et al. "I wouldn't say offensive but.": Disability-centered perspectives on large language models. In *Proceedings of the 2023 ACM Conf. on Fairness, Accountability, and Transparency*. ACM (2023), 205–216.
16. Glazko, K. et al. Identifying and improving disability bias in GPT-based resume screening. In *Proceedings of the 2024 ACM Conf. on Fairness, Accountability, and Transparency*. ACM (2024), 687–700.
17. Guo, A. et al. Toward fairness in AI for people with disabilities SBG@a research roadmap. *SIGACCESS Access. Comput.* 125 (2020), 2:1.
18. Gurari, D. et al. VizWiz grand challenge: Answering visual questions from blind people. arXiv (2018), arXiv:1802.08218.
19. Hassan, S. et al. PACE: Participatory AI for community engagement. *Proceedings of the AAAI Conf. on Human Computation and Crowdsourcing 12* (Oct. 2024), 151–154.
20. Huang, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.* 43, 2 (2025), 1–55.
21. Ives-Rublee, M. and Neal, A. Eliminating barriers to employment for disabled women - Center for American Progress. Center for American Progress (2024); https://www.americanprogress.org/article/playbook-for-the-advancement-of-women-in-the-economy/eliminating-barriers-to-employment-for-disabled-women/
22. Jang, J., Moharana, P., Carrington, P., and Begel, A. "It's the only thing I can trust": Envisioning large language model use by autistic workers for communication assistance. In *Proceedings of the 2024 CHI Conf. on Human Factors in Computing Systems*. ACM (2024), 1–18.
23. Kellogg, K., Valentine, M.A., and Christin, A. Algorithms at work: The new contested terrain of control. *The Academy of Management Annals* 14, 1 (2020), 366–410.
24. Luria, M. et al. Screened out: The impact of digitized hiring assessments on disabled workers. Center for Democracy and Technology (2024); https://cdt.org/insights/screened-out-the-impact-of-digitized-hiring-assessments-on-disabled-workers/
25. Mack, K.A. et al. "They only care to show us the wheelchair": Disability representation in text-to-image AI models. In *Proceedings of the 2024 CHI Conf. on Human Factors in Computing Systems*. ACM (2024), 1–23.
26. Makati, T., Tigwell, G.W., and Shinohara, K. The promise and pitfalls of web accessibility overlays for blind and low vision users. In *Proceedings of the 26th Intern. ACM SIGACCESS Conf. on Computers and Accessibility*. ACM (2024), 1–12.
27. Mowar, P. et al. CodeA11y: Making AI coding assistants useful for accessible web development. arXiv (2025), arXiv:2502.10884.
28. OECD. Disability, work and inclusion: Mainstreaming in all policies and practices. OECD Publishing (2022); 10.1787/1eaa5e9c-en
29. Park, J.S., Bragg, D., Kamar, E., and Morris, M.R. Designing an online infrastructure for collecting AI data from people with disabilities. In *Proceedings of the 2021 ACM Conf. on Fairness, Accountability, and Transparency*. ACM (2021), 52–63.
30. Raji, I.D. et al. Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 Conf. on Fairness, Accountability, and Transparency* (2020), 33–44.
31. Shah, M. Examining the deceptive practices of accessibility overlay tools. The Pranava Institute (2023); https://www.design.pranavainstitute.com/post/examining-the-deceptive-practices-of-accessibility-overlay-tools
32. Shestakofsky, B. Working algorithms: Software automation and the future of work. *Work and Occupations* 44, 4 (Nov. 2017), 376–423.
33. Silverman, M.A. et al. Empowering or excluding: New research and principles for inclusive AI. American Foundation for the Blind (2025); https://www.afb.org/research-and-initiatives/empowering-or-excluding
34. Silverman, M.A. et al. Technology and accommodations: Employment experiences of U.S. adults who are blind, have low vision, or are deafblind. American Foundation for the Blind (2022); https://www.afb.org/research-and-initiatives/employment/workplace-tech-study
35. Stangl, A. et al. Privacy concerns for visual assistance technologies. *ACM Trans. Access. Comput.* 15, 2 (May 2022), 15:1–15:43.
36. Tang, X., Abdolrahmani, A., Gergle, D., and Piper, A.M. Everyday uncertainty: How blind people use GenAI tools for information access. In *Proceedings of the CHI Conf. on Human Factors in Computing Systems*. ACM (2025).
37. Trewin, S. et al. Considerations for AI fairness for people with disabilities. *AI Matters 5*, 3 (2019), 40–63.
38. Wang, X. and Ahmed, S. Bodily harms: How AI and biometrics curtail human rights. Access Now (2023); https://www.accessnow.org/bodily-harms-how-ai-and-biometrics-curtail-human-rights/
39. Yadkori, Y.A. et al. Mitigating LLM hallucinations via conformal abstention. arXiv (2024), arXiv:2405.01563.
40. Zhang, X. et al. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conf. on Human Factors in Computing Systems*. ACM (2021), 1–15.

Anne Marie Piper (ampiper@uci.edu) is a professor in informatics at the University of California Irvine, Irvine, CA, USA.

Darren Gergle is the Bao Family Professor of Human-Computer Interaction in Communication Studies and Computer Science (by courtesy) at Northwestern University, Evanston, IL, USA.

Cynthia L. Bennett is a staff research scientist at Google Inc., New York, NY, USA.

Maitreya Shah is the technology policy director at American Association of People with Disabilities, Washington, DC, USA.

Chancey Fleet is an assistive technology coordinator at the New York Public Library, New York, NY, USA.

Ali Abdolrahmani is a technology accessibility specialist at the Maryland State Department of Education, Baltimore, MD, USA.

Ariana Aboulafia is the project lead for the Disability Rights in Technology Policy Project at the Center for Democracy and Technology, Washington, DC, USA.

Stacy M. Branham is an associate professor in informatics at the University of California Irvine, Irvine, CA, USA.

Sharon Koppman is an associate professor in organization and management at the University of California Irvine, Irvine, CA, USA

Arielle Silverman is the director of research at the American Foundation for the Blind, Arlington, VA, USA.

Erik B. Sudderth is a professor in computer science and statistics at the University of California Irvine, Irvine, CA, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/accessible-work

DOI:10.1145/3797488

A real-world field study examines how GitHub Copilot affects engineering productivity in an enterprise setting.

BY FEI LIU AND TODD MCKINNON

Beyond the Hype: The Efficiency-Throughput Gap with GitHub Copilot

THE RAPID ADOPTION of AI-assisted tools in software development has rekindled interest in engineering productivity. However, a holistic understanding of these tools' real-world impact on engineering productivity remains caught between marketing claims of substantial gains[1,2,22] and initial studies focused on task-level improvements.[5,8,10,20,23] While the marketing narrative is challenged by emerging real-world evidence,[11] the community still lacks deep, non-anecdotal studies that not only observe but also explain the disconnect. In this article, we aim to do just that. We introduce and investigate the *efficiency-throughput gap*, a phenomenon where individual developer efficiency gains fail to translate into measurable organizational throughput. Grounded in the Organizational Ohm's Law (OOL) framework, our real-world field study provides a methodology to both empirically document this gap and offer a data-driven explanation.

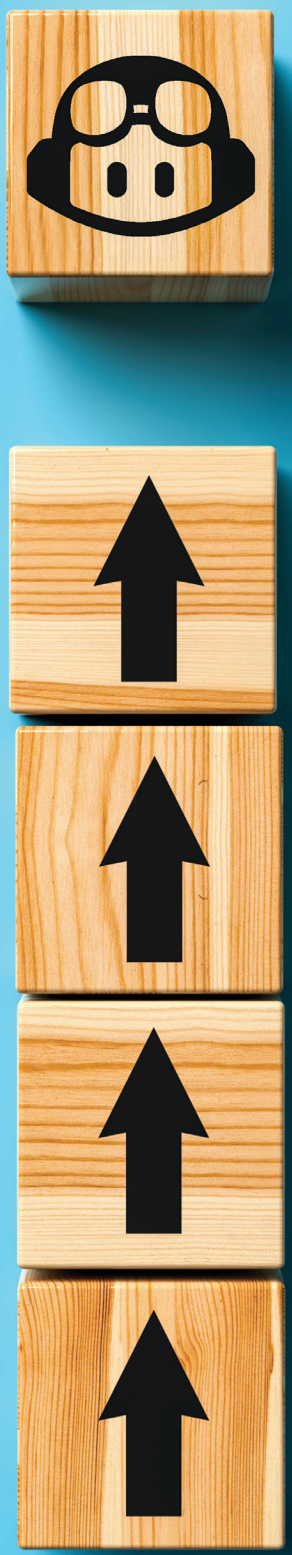
Research motivation and principles. The motivation of this work is to understand the impact of GitHub Copilot on engineering productivity to inform the strategic integration of generative AI into existing product development. Specifically, we aim to understand how engineers use GitHub Copilot, and whether its adoption yields measurable engineering productivity gains. Furthermore, this research seeks to identify the requirements for effective measurement of engineering productivity in corporate settings.

To facilitate practical business decisions, our field study was designed to reduce disruption to established engineering practices and to reflect authentic real-world conditions rather than artificial laboratory settings. Our experimental design adheres to three

» key insights

- **Despite increased engineer motivation and perceived skills, as well as reduced working hours, GitHub Copilot adoption revealed no statistically significant improvement in engineering output metrics. We call this disconnect the *efficiency-throughput gap*.**
- **The Organizational Ohm's Law (OOL) provides a practical and holistic framework for measuring engineering productivity, integrating subjective surveys with objective metrics to assess AI's impact as a function of individual potential and organizational resistance.**
- **Achieving organizational productivity gains requires more than deploying an AI-powered tool. It demands a deliberate effort to learn, adapt, and reinvent teams, processes, and infrastructure to align with AI-augmented developer capabilities.**

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING PHOTO BY MICHAELVBG/SHUTTERSTOCK



guiding principles: first, to generate actionable business insights while satisfying academic curiosity; second, to minimize the burden on participating engineers and avoid introducing unnecessary research-only processes; and third, to measure relevant data and behaviors as accurately as possible to reflect real-life operational contexts.

The act of measurement can inherently alter the state of a system, a phenomenon observed across diverse domains, from electrical circuits to quantum mechanics. This effect is particularly pronounced in social and management contexts, where the establishment of specific metrics can influence behavior, potentially leading to tactical actions or even the gaming of systems to achieve favorable results.[6,13] Recognizing the importance of minimizing such reactive effects and obtaining data reflective of genuine practices, throughout our study we prioritized transparency with participants regarding data-collection objectives, data linkage, and confidentiality. To further mitigate behavioral changes induced by the study, and unlike many observational studies that actively monitor participant activities prospectively, our primary analysis relied mainly on historical engineering metric data from recruited participants.

Organizational Ohm's Law. The study's experimental design and methodology are informed by the OOL framework, a concept we previously developed to analyze engineering productivity.[15,16] The framework draws an analogy between electrical circuits and organizational systems, stating that outcome current, and hence organizational productivity, is proportional to outcome-output efficiency and organizational potential, and inversely proportional to organizational resistance. By approximating organizational resistance through quantifiable time allocations, the law shows that engineering productivity is positively correlated with the percentage time spent on core software engineering activities (i.e., coding and testing), the relative average engineer motivation, and the relative average engineer skill levels.

Leveraging this framework, we measured engineer time allocation before and after GitHub Copilot adoption across various areas, including coding, testing, communication, waiting for decisions and dependencies, documentation, engineering processes, company processes, and inefficiency of engineering infrastructure,[14,21] alongside relative employee motivation[12,17,18] and skill ratings. The data allowed us to comprehensively assess the impact of GitHub Copilot adoption using a method grounded in a theoretically sound and intuitively sensible framework.

Despite observing improvements in engineer motivation and perceived skills, as well as reduced working hours, we found no immediate increase in key engineering metrics such as monthly pull requests and lines of code.

Our contribution. First, we present a holistic, real-world enterprise assessment of GitHub Copilot's impact, identifying and providing empirical evidence for the efficiency-throughput gap. Our findings challenge the prevailing narrative of instant AI productivity gains: Despite observing improvements in engineer motivation and perceived skills, as well as reduced working hours, we found no immediate increase in key engineering metrics such as monthly pull requests and lines of code. This result underscores that effectively realizing the potential of AI-powered tools requires continued innovation and optimization of software engineering operations. Second, we introduce a multi-method framework for holistically assessing the impact of AI tools in an enterprise setting. By integrating the OOL framework, our human-centric survey, objective engineering metrics, and participant information, our study provides a methodology that organizations can adapt to continuously evaluate the impact of AI.

Our Approach

Experimental design. This field study was conducted within the engineering organization of Okta, Inc., a publicly traded U.S. enterprise software company providing cloud-native identity and access management for modern security. Aiming for research with direct practical impact, we integrated our study with the GitHub Copilot rollout within our company. This real-world context facilitated a comprehensive evaluation of Copilot's effect on multiple engineering teams using a multi-method quantitative design. This approach involved triangulating

data from subjective Copilot productivity surveys, objective GitHub Copilot and engineering metrics, and participant background information.

To control for confounding variables from other AI tools, we leveraged a preexisting policy that discouraged exposing company code to external, non-sanctioned LLMs. A preliminary survey before the rollout confirmed this policy's effectiveness, reflecting minimal use of other AI tools and giving us high confidence in our baseline measurements (see Appendix A.1 for details). As GitHub Copilot was the only AI-powered developer tool officially deployed at scale during the study, we can attribute the observed changes primarily to its introduction.

Engineering productivity survey. To understand GitHub Copilot's impact on engineering productivity following its initial organizational rollout, we conducted a survey several months after the GitHub Copilot rollout using Qualtrics, a commercial Web-based survey platform. The survey instrument gathered data on participants' background, their average weekly time allocation across various work activities, Likert-scale ratings of their motivation and skills, Likert-scale assessments of code quality and test frequency both before and after Copilot usage, the frequency of Copilot utilization, areas of perceived Copilot usefulness, a Likert-scale rating of Copilot's perceived productivity enhancement, concerns regarding its use, and open-ended comments on engineering productivity or GitHub Copilot experiences (see Figure A1 in the appendix for an example of the survey instrument). The survey was designed to be brief, requiring approximately five minutes to complete.

Participant recruitment was conducted in partnership with various engineering teams as part of their GitHub Copilot deployment. Between November 18 and December 20, 2024, we used Slack and email to invite 261 individuals who had been granted Copilot licenses between July and September 2024 (see Figure A2 in the appendix for the recruitment message). We received 97 responses, 95 of which included email addresses, enabling us to link their survey data with their GitHub Copilot and engineering metrics. Participation in the survey was voluntary and uncompensated. (The distribution of professional experience among survey respondents is shown in Figure A3 in the appendix.)

GitHub Copilot and engineering metrics. Transparency regarding data linkage was maintained throughout the study. Both the survey instrument and participant recruitment communications (via Slack and email) explicitly stated that participants' email addresses would be used solely to correlate survey responses with GitHub Copilot and engineering metrics for this research on engineering productivity. Participants were also assured that all data would be analyzed anonymously and kept confidential. Consequently, we used participants' email addresses as unique identifiers to link their subjective survey input with objective metrics.

We extracted relevant metrics using Faros.ai,[a] a commercial engineering data platform that integrates with our company's suite of engineering software applications (such as GitHub and JIRA) via representational state transfer APIs (REST APIs). Faros.ai transforms the retrieved data into its proprietary schema, enabling us to query and extract key metrics through its dashboards. The metrics included Copilot activation date, Copilot usage statistics, pull request (PR) count, PR comment count, PR review count, and lines of code (LOC) contributed. To establish a before-Copilot baseline, we used data from April, May, and June 2024. After-Copilot data was represented by the period of September, October, and November 2024.

Participant information. The engineering productivity survey included questions about participants' engineering experience, capturing variables such as years of experience in software development and their primary programming language. To provide further context, we enriched the survey data with information from our company's Workday system, including variables such as hire date, job title, job level, job location, and engineering team.

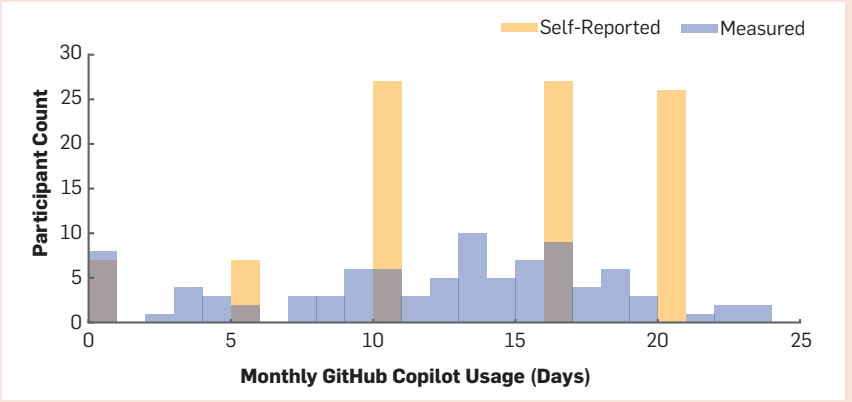
Data preparation. *Data cleaning.* The collected datasets underwent a data cleaning and preprocessing phase to address potential errors and inconsistencies across both the survey responses and metrics data. This process involved correcting participant identifiers, verifying and adjusting anomalies in self-reported data, and handling incomplete responses. We adopted a data strategy of retaining all responses and metric data for further analysis. While this necessitates more detailed processing and analytics, it allows us to maximize our dataset, which is crucial given our moderate sample sizes. More importantly, this approach more accurately reflects the complexities of real-world engineering productivity measurement, identifying edge cases and addressing them through subsequent analyses. This resulted in a final dataset of 95 valid responses, of which 74 were complete for the before-and-after analyses. (See Appendix A.3 for a detailed description of the data-cleaning procedures.)

The raw GitHub Copilot and engineering metrics, extracted monthly, exhibited inherent variability. To mitigate these fluctuations and account for onboarding and departures, we employed a three-month rolling average for both the before-Copilot (April to June 2024) and after-Copilot (September to November 2024) periods. We then calculated the number of valid monthly data points for each participant and computed the mean and standard deviation based on these valid months. The analysis comparing the three-month average after-Copilot metrics with metrics calculated only for the months after individual Copilot license assignment yielded similar conclusions. Consequently, the subsequent analysis used the three-month average engineering metric data from September to November 2024 to ensure consistency and reduce variability.

Data integration. To integrate data obtained from multiple sources and systems, we used participants' email addresses as a unique identifier for record linkage. Data cleaning, joining procedures, and insight analyses were implemented using Python within Jupyter Notebook. Following data preparation, we generated a unified dataset in CSV format, encompassing the survey responses, GitHub Copi-

a https://www.faros.ai/

Figure 1. Histograms of self-reported and measured monthly GitHub Copilot usage.



lot and engineering metrics, and enriched participant information. This integrated dataset comprised up to 150 direct and derived attributes for each participant, enabling multifaceted analyses of engineering productivity for survey participants. For clarity, we focus on the analyses and results related to GitHub Copilot's impact.

Assessing the Impact of GitHub Copilot

GitHub Copilot utilization. The survey included a question asking participants about their recent GitHub Copilot usage: "How often have you used GitHub Copilot recently?" We also collected objective GitHub Copilot usage metrics for the same period. Our analysis showed substantial variability in Copilot usage among participants, with 7% reporting no usage during the three months following Copilot deployment, illustrated in Figure 1. To assess the consistency between self-reported (orange bars) and measured (blue bars) usage, we compared these two datasets. Despite inherent differences in measurement methodology, data source, and data type, statistical tests indicated a statistically significant ($p < 0.001$), medium-to-large (Pearson's $r = 0.48$, Spearman's $\rho = 0.44$) correlation,[9] suggesting a general agreement between perceived and actual Copilot usage behavior among the participants. In essence, this comparison acts as a quality check that strengthens the foundation of the entire study, allowing us to proceed with the analysis of the other self-reported survey questions with a greater degree of confidence in the data's integrity.

To evaluate potential survey participation bias, we also compared the objective Copilot usage metrics of survey participants with those of all licensed Copilot users within our company. This comparison showed that survey participants were statistically significantly more likely to use Copilot and exhibited a higher frequency of usage. Consequently, the findings presented in this work may reflect a more optimistic assessment of GitHub Copilot's impact than the actual experience across the entire user base.

Before vs. after GitHub Copilot comparison. To determine the appropriate statistical tests for evaluating paired before-and-after data related to the GitHub Copilot survey response and engineering metrics, we implemented the following data analysis. Following a Shapiro-Wilk normality test and a symmetry check (mean absolute deviation from median $\leq$ standard deviation), one of three statistical tests was applied to the paired differences of Copilot users:

- If the differences passed the normality test ($p > 0.05$), we applied the paired t-test. The magnitude of the change was measured using Cohen's d as the effect size.
- If the differences were not normally distributed but were symmetric, we used the Wilcoxon signed-rank test. The magnitude of the change was measured using rank-biserial correlation (r) as the effect size.
- If the differences were not symmetric, we applied a sign test, implemented as a two-sided binomial test on the count of positive vs. negative differences. The magnitude of the change was measured using Cohen's g as the effect size.

The results of these statistical tests are visualized in Figures 2 and 3. For each response or metric, the corresponding subplot displays the before-and-after means, the type of statistical test performed, the resulting p-value, the effect size test, and the corresponding value, as well as the data pair count. If the selected statistical test yielded a statistically significant result ($p \leq 0.05$), the before-and-after bars in the subplot are colored green; otherwise, they are colored pink.

Engineer time allocation. To understand engineer time allocation, we categorized engineering activities into several distinct categories and asked the survey participants to report their weekly time spent on each. Figure 2 (a–j) illustrates the mean reported time allocation and corresponding 95% confidence intervals before and after GitHub Copilot usage. Following Copilot adoption, the participants reported a reduction in the average time spent across all activity categories. The largest time savings were reported in coding (1.06 hours per week) and testing (1.56 hours per week). Overall, participants reported an improved work-life balance, with average weekly work hours decreasing from 44.77 to 40.83. Statistically significant reductions ($p \leq 0.05$) were observed in time spent on context shifting, coding, documentation, testing, and total working hours. The effect sizes for these reductions were large, with rank-biserial r ranging from -0.51 to -0.90,[9] indicating a meaningful impact. In contrast, time allocated to more human-centric areas, such as communication, waiting for decisions and dependencies, company processes, engineering processes, and inefficiency of engineering infrastructure, did not show statistically significant changes.

The data also illustrated that approximately 30% of participants' work time was allocated to coding, 10–15% to testing, and slightly more than 50% to other activities. An analysis of the work time allocation by job level showed a statistically significant dependency (analysis of variance (ANOVA), $p \leq 0.05$). Junior engineers reported spending 50–70% of their time on coding and testing, while senior engi-

neers allocated less than 30% to these tasks (see Figure A4 in the appendix). This pattern aligns with typical expectations for engineering career progression, where more senior roles involve leading initiatives, contributing to technical strategy, cross-organizational collaboration, and mentorship, rather than direct code writing and testing.

Engineer motivation and skills. Figure 2 (k–l) presents the before and after ratings for engineer motivation and skills with 95% confidence intervals. Participants reported statistically significant improvements in both their motivation and perceived skill levels after adopting GitHub Copilot. These improvements were accompanied by very large effect sizes (rank-biserial r of 0.95 for motivation and 1.00 for skills).[9]

Pull requests. Figure 3 (a–h) presents the before and after three-month average monthly PR count, PR cycle time, monthly PR comment count (comments received), monthly PR review count (reviews performed), and their corresponding 95% confidence intervals. Contrary to findings in existing research and marketing, and perhaps unexpectedly, only the monthly PR review count demonstrated a statistically significant increase; the other PR metrics showed no significant improvements. While we treated each PR equally in our analysis of the monthly PR counts, it is important to acknowledge that PRs can vary significantly in effort. Therefore, incorporating proper PR metadata, such as PR sizing, in future analyses could further improve the representation of engineering output.

Monthly lines of code. Monthly LOC metrics were calculated as the mean of the three-month average LOC for each participant. We extracted several LOC variants, including lines added, lines deleted, diff size, and net lines added. In practice, raw LOC proved an unreliable proxy for individual coding effort, with before Copilot averages exceeding an implausibly high 30,000 lines added per month. Analysis of individual contributions uncovered that the LOC consisted of diverse code development activities beyond feature coding. For example, feature development teams may primarily engage in writing application logic (the type of coding often studied in controlled environments); engineering platform teams may generate extensive configuration code with minimal manual input; and tech debt teams can produce large LOC changes during refactoring.

To isolate LOC indicative of direct engineering authoring, we considered focusing solely on feature development teams or repositories. However, this approach was insufficient due to the colocation of feature code with dependency, configuration, and metadata code within the same repositories, often without consistent directory or naming conventions that would allow for reliable automated separation. As an alternative, we explored filtering PRs based on the magnitude of LOC changes, hypothesizing that very large changes were less likely to represent direct, line-by-line coding

Figure 2. Comparison of engineer time allocation, motivation, and skills before and after GitHub Copilot for Copilot users.

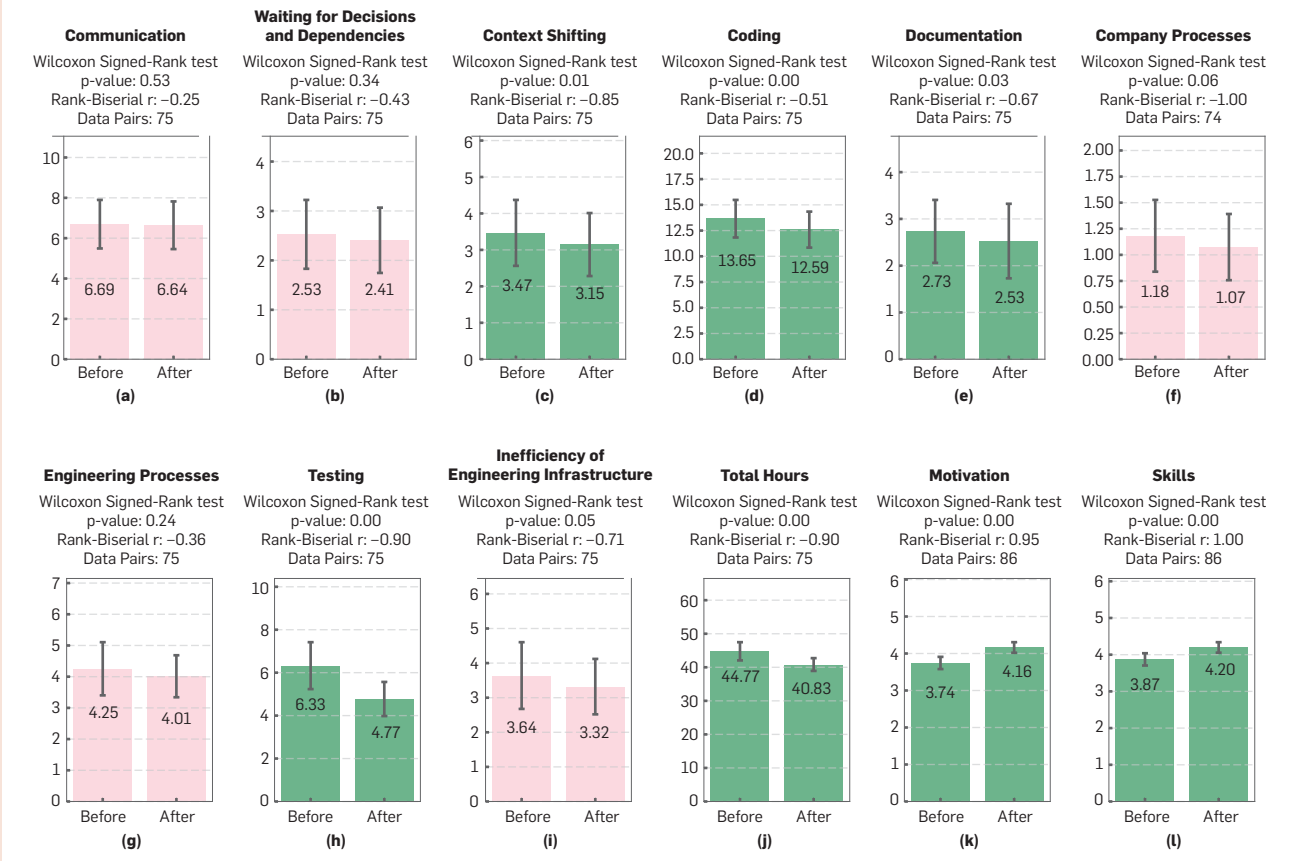
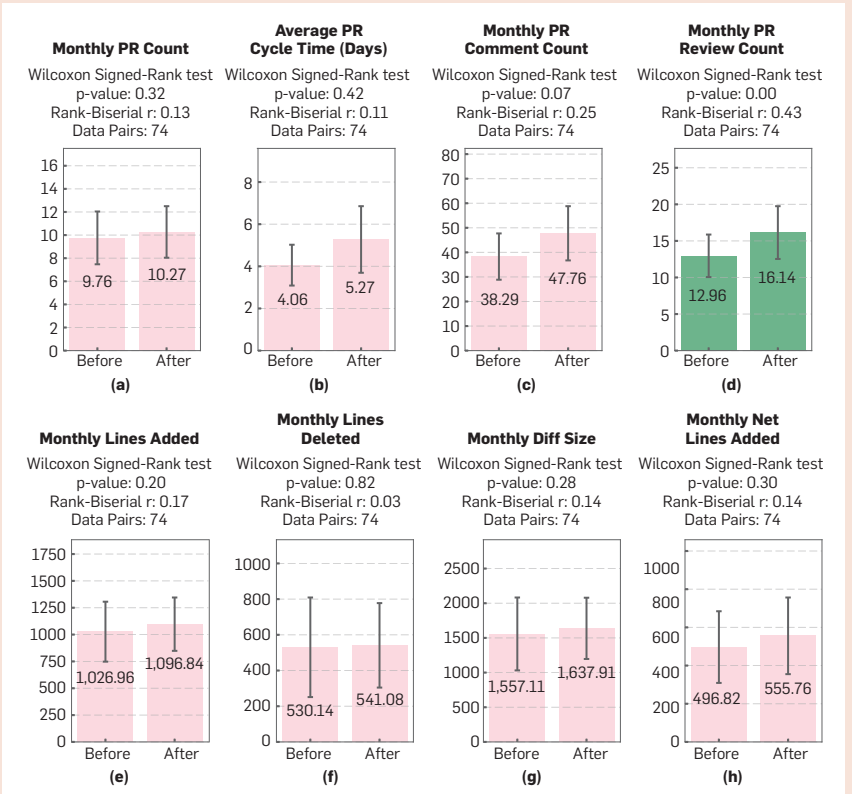


Figure 3. Comparison of engineering metrics before and after GitHub Copilot for Copilot users.



effort. Drawing upon industry best practices for PR reviews and interviews with our engineers regarding common organizational practices, we developed a heuristic rule to identify PRs potentially containing engineering-authored code: (Lines added and deleted < 801) OR (Lines added and deleted < 1001 with > 5 comments) OR (Lines added and deleted < 3001 with > 7 comments) OR (Lines added and deleted < 5001 with > 10 comments).[3,4,7] LOC contributions from PRs not meeting this heuristic were excluded from participants' monthly LOC totals. These recalculated LOC metrics values were not overly sensitive to variations in the heuristic rule. Moreover, recalculating the various LOC metrics after this filtering yielded values within a more reasonable range for direct coding effort,[16,19] as shown in Figure 3 (e–h). Nevertheless, consistent with our findings for PR metrics, neither the raw LOC metric variants nor the recalculated ones showed statistically significant changes following Copilot adoption.

Additionally, for participants who were assigned Copilot licenses but did not use them, we observed no reported changes in survey responses and no statistically significant changes in any engineering metrics, as shown in Figures A5 and A6 in the appendix. However, the small sample size of this non-user group suggests that these null findings should be interpreted with caution.

Pareto insights. We also investigated the hypothesis, consistent with existing literature,[20] that AI tools benefit less experienced engineers more. The goal of this analysis was to see if Copilot led to a more even distribution of work output. To test this, we conducted a Pareto analysis of PR count before and after GitHub Copilot for participants who used it (Figure A7). Prior to Copilot, 46% (34 out of 74) of participants contributed to 80% of the total PR output. Following Copilot implementation, this proportion increased to 53% (39 out of 74) of participants, accounting for 80% of the slightly increased total PR output. The observed percentage shift implies that Copilot may have led to a more even distribution of PR contributions among engineers, potentially benefiting those with lower initial PR output. However, statistical analysis found that the difference between the before and after distributions was not statistically significant.

GitHub Copilot's benefits and concerns. To assess GitHub Copilot's benefits and concerns, we surveyed participants with multiple-choice questions. The first, "Which areas do you find GitHub Copilot useful?" offered categories aligned with engineer time allocation, motivation, and skills, including a detailed breakdown of coding tasks from simple to complex (Figure 4a). Participants reported Copilot to be most useful for boilerplate code, function templates, queries, configuration files, and testing. They found it less capable, however, of developing front-end components and back-end services, writing complex logic or algorithms, and supporting human-centric enterprise development activities. The second question, "What are the reasons you don't use GitHub Copilot after being assigned a seat?" highlighted that the primary reasons for non-use were concerns about Copilot's accuracy and lack of time and knowledge to use Copilot (Figure 4b). Despite the limitations, 94 participants rated GitHub Copilot's impact on overall productivity at an average of 3.95 out of 5 on a Likert scale, indicating a reasonably positive perception.

We then sought to validate whether the self-reported time reductions observed in activities like coding and testing (Figure 2) were statistically linked to these perceived benefits. Our hypothesis was that engineers who perceived Copilot as useful for a specific activity (e.g., "testing") would be the same engineers who reported a time reduction for that activity. To test the hypothesis, for each engineer time-allocation activity, we calculated both value and binary Pearson correlations, along with their corresponding p-values, between time-allocation changes and GitHub Copilot benefit mentions, to test the basic association. Additionally, we performed a Chi-square test (χ^2 test) to assess the association between binary time-allocation changes and benefit mentions. Finally, we used a two-sample t-test to compare the mean time-allocation changes be-

tween engineers who mentioned Copilot benefits for a specific activity and those who did not.

Our analyses demonstrated relatively strong negative value correlations between time-allocation changes and benefit mentions for context shifting, engineering process, testing, and inefficiency of engineering infrastructure. Relatively strong binary correlations were found for context shifting, coding, documentation, engineering process, testing, and inefficiency of engineering infrastructure. The Chi-square test indicated a significant association between the binary changes in context shifting, coding, documentation, engineering process, testing, and inefficiency of engineering infrastructure, and their corresponding benefit mentions. Furthermore, the t-test showed significant differences in the mean metric change values of context shifting, coding, engineering process, and testing between the benefit mentioned and not-mentioned groups (see Appendix Table A1 for details).

Collectively, these analyses suggest a reasonable relationship between the four time-allocation activities that showed statistically significant reductions before and after GitHub Copilot adoption (context shifting, coding, documentation, and testing) and the perceived benefits of Copilot.

Discussion

This work provides a multifaceted examination of GitHub Copilot adoption, yielding insights into software productivity measurement and the tool's impact on enterprise software development. On the one hand, we demonstrate the utility of OOL as a practical and holistic framework for understanding engineering productivity by integrating engineer activity (time allocation), mental state (motivation), and capability (skills) alongside engineering metrics. On the other hand, our real-world investigation highlights the inherent gaps in accurately measuring engineering productivity, suggesting the potential value of more rigorous practices, such as consistent repository directory labeling, pull-request sizing, and the identification of AI-generated code to improve future assessments.

Regarding GitHub Copilot's impact, our findings reveal a critical distinction between individual efficiency and organizational throughput. While GitHub Copilot usage varied significantly among engineers, the tool demonstrably fostered positive changes in perceived engineer value, reduced time spent on various engineering activities, and boosted motivation and perceived skills. Yet we observed no immediate direct correlation with gains in key engineering metrics. This resulted in an individual efficiency gain (the same work in less time) but no statistically significant organizational throughput gain (no increase in monthly PR count or lines of code). According to the OOL framework, we hypothesize that productivity improvements from such tools occur in three stages:

1. *Efficiency gains:* The tool boosts average individual potential (motivation, skills) and reduces resistance on specific tasks such as coding and testing. However, the overall outcome current remains unchanged. The efficiency gains benefit individuals with a better work-life balance. Our data suggests early signs of this first stage, but not yet the second.

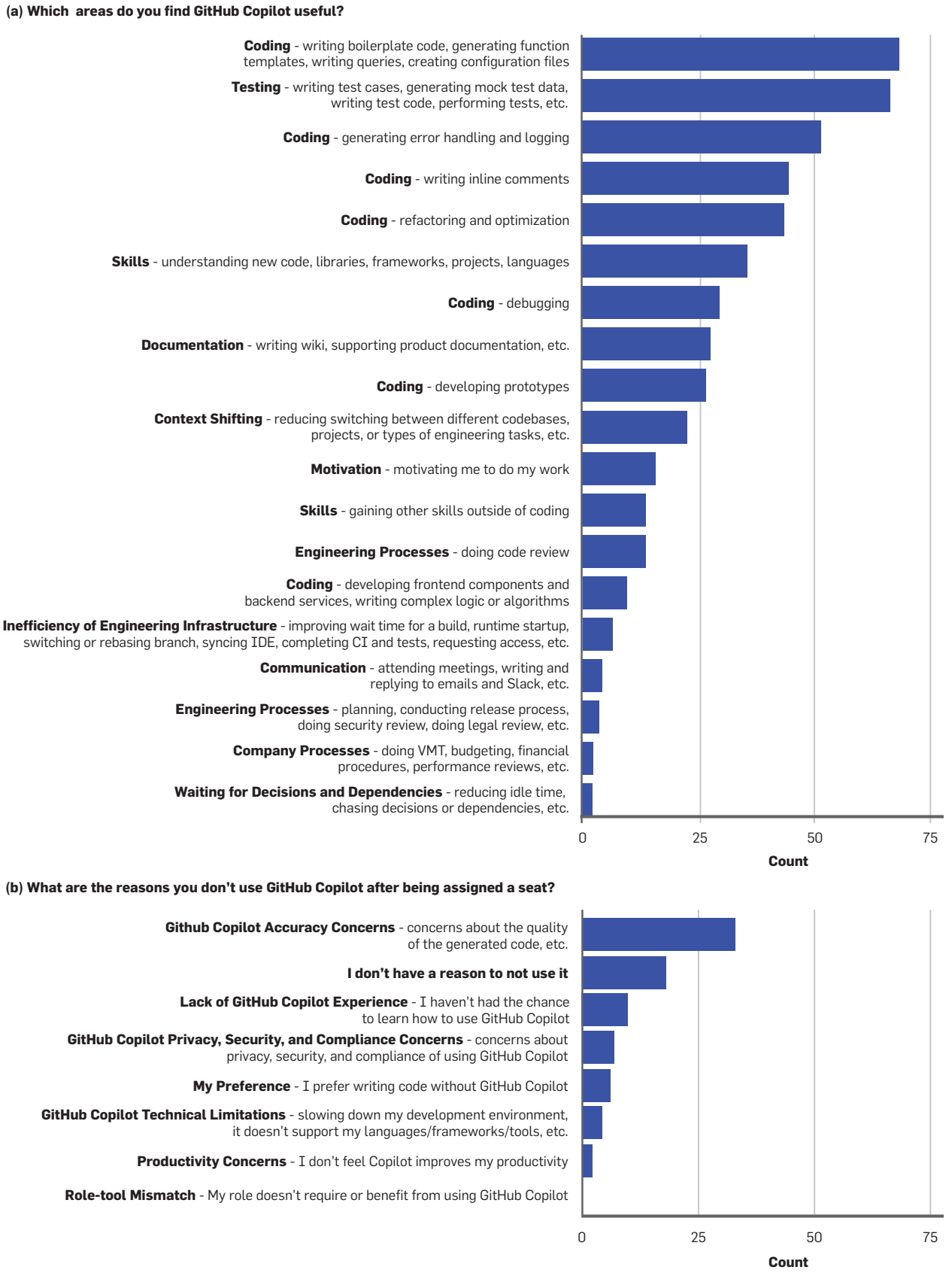
2. *Productivity gains:* To increase the outcome current, the organization must consciously reduce the total resistance and further increase individual potential. This requires upskilling developers to leverage the tool's functionalities, adjusting productivity expectations, and optimizing development workflows to align with AI-augmented developer capabilities.

3. *Organizational optimization:* The organization can redesign its organizational circuit to lower human-centric resistance. This involves structural changes, such as creating more autonomous teams or increasing individual scope to reduce the overhead from communication and decision making, as well as exploring effective modes of human-AI teaming.

This work highlights that to truly achieve organizational productivity gains, deploying an AI-powered tool is not enough. It requires a deliberate effort from engineering and management to adapt, learn what the tool can and cannot do for their specific problems, as the tool's functionalities evolve quickly, and reinvent teams, engineering processes, and infrastructure to

For future work, we recommend prioritizing longitudinal studies across diverse organizations to examine the long-term effects of AI-powered tools on engineering productivity.

Figure 4. Engineering perspectives on GitHub Copilot: (a) Benefits (n=92) and (b) Concerns (n=79).



more online

To access the appendix for this article, please visit https://dl.acm.org/doi/10.1145/3797488 and click on Supplemental Material.

capture the potential efficiencies. The findings informed our organization's post-study strategy. Recognizing the strong positive impact on individuals, we accelerated Copilot adoption by removing management approval while simultaneously monitoring usage. We began sharing best practices to improve tool utilization, building a shared understanding of its current constraints, and identifying new challenges. Furthermore, we are embedding AI impact measurement into our regular engineering process and introducing a broader range of AI-powered tools, such as Copilot agent mode and various vibe-coding applications. The endeavor represents the beginning of a co-invention of the tool, the team, the process, and the infrastructure required to fully explore and realize the potential of AI-driven productivity gains.

While this study provides comprehensive analyses and insightful findings, its scope is subject to several limitations. First, the research was conducted within a single company and involved a moderate number of participants, potentially limiting the generalizability of the results. Furthermore, the sample size means some observations of statistical insignificance could be due to a lack of statistical power rather than an absence of effect. Second, as the study captured the initial rollout phase of GitHub Copilot, the observed impacts may not fully reflect its longer-term effects. However, our subsequent monitoring of PRs and LOC for participants from December 2024 to May 2025 showed no statistical improvements, supporting the point that time alone is insufficient to translate individual efficiency into organizational throughput. Third, the objective output metrics we used (e.g., monthly PR count and monthly LOC) are, at best, proxies for engineering productivity and do not capture the full picture of engineering output. Fourth, this work primarily focused on the impact on engineering output metrics, rather than broader business outcomes such as accelerated time to market or product quality improvements. Fifth, the analysis was based on the organization's adoption of GitHub Copilot's AI-assisted coding functionalities available at the time, including autocomplete and chat. Given the rapid evolution of AI-powered development tools, the implications of more advanced capabilities, such as retrieval-augmented generation (RAG) over enterprise codebases or deeper engineering workflow integrations, warrant separate investigation. Ongoing advances in these AI-powered tools could alter their impact. For future work, we recommend prioritizing longitudinal studies across diverse organizations to examine the long-term effects of AI-powered tools on engineering productivity, thereby providing valuable technical, business, and economic insights for researchers and practitioners.

Summary

This field study of GitHub Copilot productivity impact moves beyond the hype of instant gains by identifying an efficiency-throughput gap, revealing a more nuanced reality than that often presented in existing research and marketing. Our findings underscore the challenges that organizations face in translating AI-driven individual efficiency gains, which we observed as improved engineer sentiment and a reduction in working hours, into immediate and quantifiable organizational productivity throughput, where we found no statistically significant improvement in output metrics. By integrating the Organizational Ohm's Law framework, our human-centric survey, and objective engineering metrics, this study also provides a robust and repeatable methodology. It offers a practical toolkit for organizations to move beyond one-time snapshots toward a continuous assessment process of AI's evolving impact, paving the way for iterative co-invention and, ultimately, tangible engineering productivity gains.

Acknowledgments

We thank Bhawna Singh, JC Tierney, Robert Lucero, Demi Dina, Pablo Terradillos, Steve Workman, Vikas Gupta, and Chanaradee Leelamanthep for their valuable collaboration. We also extend our appreciation to Shubha Nabar and Bill Farrakh from Faros.ai for their support, and to the anonymous reviewers for their insightful feedback.

References

1. Alphabet. Alphabet 2024 Q3 earnings call transcript (2024); https://thetranscript.net/transcript/7008/alphabet-q3-2024-earnings-call-transcript
2. Alphabet. Alphabet 2025 Q1 earnings call (2025); https://abc.xyz/2025-q1-earnings-call/
3. Atlassian. The written and unwritten guide to pull requests (2019); https://www.atlassian.com/blog/git/written-unwritten-guide-pull-requests
4. Bacchelli, A. and Bird, C. Expectations, outcomes, and challenges of modern code review. In *International Conference on Software Engineering* (2013), 712–721.
5. Bakal, G. et al. Experience with GitHub Copilot for developer productivity at Zoominfo. *arXiv* (2025), arXiv:2501.13282.
6. Baker, G.P. Incentive contracts and performance measurement. *Journal of Political Economy* 100, 3 (1992), 598–614.
7. Bosu, A., Greiler, M., and Bird, C. Characteristics of useful code reviews: An empirical study at Microsoft. In *Working Conference on Mining Software Repositories* (2015), 146–156.
8. Chatterjee, S., Liu, C.L., Rowland, G., and Hogarth, T. The impact of AI tool on engineering at ANZ Bank: An empirical study on GitHub Copilot within corporate environment. *arXiv* (2024), arXiv:2402.05636.
9. Cohen, J. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates (1988).
10. Cui, K.Z. et al. The productivity effects of generative AI: Evidence from a field experiment with GitHub Copilot. *An MIT Exploration of Generative AI* (2024).
11. Denisov-Blanch, Y. Does AI actually boost developer productivity? (100k devs study). (2025); https://www.youtube.com/watch?v=tbDDYKRFjhk
12. Herzberg, F., Mausner, B., and Snyderman, B.B. *The Motivation to Work*. Wiley (1959).
13. Kerr, S. On the folly of rewarding A, while hoping for B. *Academy of Management Journal* 18, 4 (1975), 769–783.
14. Křečková, J., Kennedy, D.M., Brožová, H., and Rydval, J. Project management communication planning: An improved optimization model with additional recipients and individualized weights. *IEEE Trans. on Engineering Management* 69, 4 (2022), 1067–1080.
15. Liu, F. and McKinnon, T. Organizational Ohm's Law—Case in point: Engineering velocity. *IEEE Transactions on Engineering Management* 71, (2024), 3832–3842.
16. Liu, F. and McKinnon, T. Organizational Ohm's Law in practice: Measuring engineering productivity. *IEEE Transactions on Engineering Management* 71, (2024), 11494–11504.
17. Locke, E.A. and Latham, G.P. *A Theory of Goal Setting & Task Performance*. Prentice-Hall, Inc. (1990).
18. Maslow, A.H. A theory of human motivation. *Psychological Review* 50, 4 (1943), 370–396.
19. McConnell, S. *Software Estimation: Demystifying the Black Art*. Microsoft Press (2006).
20. Peng, S., Kalliamvakou, E., Cihon, P., and Demirer, M. The impact of AI on developer productivity: Evidence from GitHub Copilot. *arXiv* (2023), arXiv:2302.06590.
21. Xia, X. et al. Measuring program comprehension: A large-scale field study with professionals. *IEEE Trans. Software Eng.* 44, 10 (2018), 951–976.
22. Zeff, M. Microsoft CEO says up to 30% of the company's code was written by AI. *TechCrunch* (2025); https://techcrunch.com/2025/04/29/microsoft-ceo-says-up-to-30-of-the-companys-code-was-written-by-ai/.
23. Ziegler, A. et al. Measuring GitHub Copilot's impact on productivity. *Commun. ACM* 67, 3 (2024), 54–63.

Fei Liu (fei.liu@okta.com) is a principal emerging technology researcher at Okta, Inc., San Francisco, CA, USA.

Todd McKinnon is the cofounder and CEO at Okta, Inc., San Francisco, CA, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

Watch the authors discuss this work in the exclusive *Communications* video. https://cacm.acm.org/videos/beyond-the-hype

DOI:10.1145/3800692

Results from a 22-institution, multi-year project to broaden participation in computing show that implementing even a subset of the 10 systemic changes leads to increases in retention and attraction.

BY CARLA E. BRODLEY, CATHERINE GILL, MEGAN GIORDANO, AND CAITLIN J. KIDDER

10 Systemic Changes to Broaden Participation in Computing

THE CENTER FOR INCLUSIVE COMPUTING (CIC) was established in 2019 at Northeastern University to broaden participation in computing (BPC) at U.S. universities. BPC has been a goal of the U.S. computer science (CS) community for several decades, and much work has been done to design and implement best practices. But most efforts reported in the literature explore the efficacy of BPC activities at a single institution or a handful of institutions. Further, progress in BPC nationwide has been slow; in 2019, students who identified as women and/or from races and ethnicities historically marginalized in tech continued to be underrepresented in computing relative to their representation in higher ed.[4] The 2018 report "Rebooting Representation"[33] stated that although there appeared to be consensus on BPC best practices, uptake was slow. The report called attention to this "paradox," hypothesizing that university computing departments lacked sufficient incentives and resources to change. Building on previous research and with core funding from Pivotal, the CIC set out to determine which BPC practices work *across* different university contexts. The CIC funds changes that create a more inclusive environment for undergraduates. By *inclusive environment,* we mean one in which all students—including those who have never programmed—feel invited to try computing and supported to pursue a major in the field.

At the CIC, we work with the largest U.S. universities[a] to reach as many undergraduates as possible. Our ap-

a We work with nonprofit universities that produce 150 or more CIP 11 (computing-related majors) graduates annually. In the U.S., the Classification of Instructional Programs (CIP) provides a taxonomic scheme that supports the tracking and reporting of fields of study and program completion activity.[30] CIP code 11 is for computing majors and includes CS.

» key insights

- **The Center for Inclusive Computing (CIC) launched in 2019 to partner with U.S. university computing departments to identify and remove barriers that prevent students from discovering and thriving in computing. We were given the unique opportunity to determine which interventions are most effective in removing these barriers, and to do so at a previously unheard-of scale.**
- **We report results across 22 universities of the 10 systemic changes we have found most impactful and we describe how they can be adapted to different institutional contexts.**
- **The results show that implementing even a subset of these changes leads to increases in retention and attraction across all populations.**

IMAGE BY ANDRIJ BORYS ASSOCIATES; PHOTOS COURTESY OF NORTHEASTERN UNIVERSITY (CIC.NORTHEASTERN.EDU)











proach seeks to test two hypotheses. The first is that, by making *systemic* changes (e.g., to the courses and/or the degree structure), more students would try and/or be retained in computing (i.e., What happens if we focus on fixing the institution rather than fixing the student?). The second hypothesis is that, by targeting *systemic* changes, much of the work would be *sustainable* (i.e., would not require ongoing fundraising or add material cost to the department's budget).

This article describes the results from Phase One of this experiment. Specifically, we share aggregate data from the first 22 schools funded for implementation at the undergraduate level. These 22 schools began implementation between 2020 and 2022, meaning enough time has elapsed for us to measure the impact. (Phase 2 will include results for more than 50 schools.) For each university, the CIC provides: multi-year *grants* of $200,000 (or more) to cover implementation costs; *technical assistance* through a *program manager* (a computing faculty member and CS-education expert); and a platform to collect and analyze the *data* to track the impact of changes. Each university's grant is evaluated annually, and the next tranche of funding is contingent upon satisfactory effort and progress against the prior year's work plan.

We begin the article by summarizing the methodology used to select the 22 schools (and schools that have joined the portfolio subsequently) and what an engagement with the CIC entails. Next, we describe 10 systemic interventions that we have found (through trial and error) to be effective in creating an inclusive environment for undergraduates, regardless of prior exposure to CS. We then map the 22 schools to the 10 interventions and share the aggregate results, including a comparison of the change in demographics of computing majors relative to the changes in overall university enrollment. Finally, we describe future plans and draw conclusions from our work.

High school CS classes are more often present in zip codes of economic privilege. The problem is that students without coding experience can quickly feel behind when, in reality, they have only just begun.

Framework for Systemic Change

The CIC's methodology has the following six steps:

1. We announce the availability of funding via our website and listservs. Per the conditions of our core funding, the initial eligibility criterion was nonprofit U.S. institutions graduating 200 or more students in IPEDS (Integrated Postsecondary Education Data System) CIP code 11.[30] We now accept applications from schools graduating 150 or more students in CIP code 11.

2. We meet online with each school to review their application and formulate an initial hypothesis regarding barriers and potential changes. We then decide whether to do an in-person site visit based on the level of leadership commitment, the relevance of the systemic interventions to the barriers discussed, and faculty buy-in.

3. If we proceed, we schedule a site visit, which is a 4–6 hour, in-person (except during COVID) set of meetings with the project team, intro sequence faculty, teaching assistants (TAs), academic advisors, students, and the dean. The purpose is to hear from all stakeholders to further diagnose the barriers and assess the potential for change.

4. We summarize the site visit in a memo that identifies key themes, connecting the findings to evidence-based interventions and pattern matching to other schools with similar institutional contexts.

5. If we choose to fund the school, we work with the project team on a work plan and budget. We now provide costing guidelines that show the budget range for each intervention.[b] The work plan and budget (which includes funding for data collection and 15% overhead) are included as exhibits in the grant contract.

6. Selected schools implement the agreed-upon changes, adjusting as needed in consultation with the CIC and their program manager, and submit annual progress reports. The funding is tranched, meaning subsequent installments are contingent upon satisfactory completion of the prior year's activities.[c]

Grantees (selected schools) submit data twice a year, including historic data for the two years prior to launch.[d]

b Please see https://cic.northeastern.edu/costing-guidelines for the average cost of each intervention.

c In three cases we have halted ongoing funding because a school was not able to make the mutually agreed upon changes.

d See full data collection description at https://cic.northeastern.edu/dataprogram/

Often, faculty and departments struggle to access intersectional enrollment and outcomes data, as this data tends to be centrally held. Our experience is that the CIC funding unlocks access.[4,27] Specifically, for each term,[e] schools submit data for CS1–3 for the following variables:

- Count of students: non-negative integer
- Racial identity: Hispanic/Latinx, any race; American Indian or Alaska Native, not Hispanic (AIAN); Asian, not Hispanic; Black or African American, not Hispanic; Native Hawaiian or Other Pacific Islander, not Hispanic (HPI); White, not Hispanic; Nonresident Alien, Race/Ethnicity Unknown
- Gender identity: gender X/U, men, women
- Major: CS,[f] other computing, non-computing, undeclared
- Transfer: Students who transferred from another institution during the current academic year
- Course outcome: pass,[g] fail, or withdraw (PFW)

CS1–3 are defined as the first three required programming courses in the CS major that are are not taken in the same term (i.e., none are discrete math). Typically these courses are: intro to programming, object-oriented programming, and data structures. If a department does not have a required third course, then CS3 is the course taken after CS2 by the majority of CS majors. For each combination of the values of these variables, schools enter the count of students (no missing data allowed), thus ensuring the data is intersectional. The time required for the initial setup and submission averages 40 hours; subsequent submissions average 8–10 hours.

Finally, we note that the data system allows schools to compare their data against 18 different peer groups of universities in the CIC's portfolio.

10 Systemic Interventions

We have come to focus on 10 interventions through a combination of extensive literature searches on systemic changes in higher ed,[21,26,35,h] observations from our first two rounds of site visits, pattern recognition *across* schools, and data analysis of the results of the first 22 universities. To home in on these 10 interventions, we started with the literature, then noted the issues that seemed to arise at all universities related to retention and discovery. Specifically, students across the schools in our first and second rounds of site visits shared a number of concerns, including: if they were new to coding, how difficult it was to be in the same class as those with prior experience; less than optimal experiences with the TAs; that different sections of the same class taught by different professors had different levels of preparation for the next class in the sequence; that class dynamics were terrible at schools with extra barriers to major declaration; challenges engaging with some assignments; and that unless they discovered CS in their first semester it was impossible to major and graduate on time. We heard many of these same concerns from TAs and advisors, and further observed that advisors were not always able to explain different pathways (for example, the difference between the B.S. and B.A. CS degrees at their own school), leading us to realize that advisors need to be kept in the loop. The first three interventions focus on changes to the intro sequence in order to attract students to and retain them in the major.[3] The next four focus on access to the degree and the degree's structure. The final three focus on co-curricular supports for students in the major.

1. Manage distribution of prior experience. CS is offered in 60% of high schools in the U.S., but less than 10% of high school students take CS.[18] As such, universities have a significant opportunity to promote the discovery of computing in college. To do that, schools need strategies for managing the distribution of prior experience.[2,5] But care must be taken because prior experience is not uniformly distributed across genders, races, and ethnicities, and high school CS classes are more often present in zip codes of economic privilege. The problem is that students without coding experience can quickly feel behind when, in reality, they have only just begun.[23,34] Indeed, students have reported CS1 classes being overly difficult and unfriendly when students with and without prior experience are in the same class.[25] Finally, from a faculty perspective, it is challenging to teach a class when students come with widely differing prior knowledge, and sometimes neither group is engaged.

In the June 2022 issue of CRA's *Computing Research News,*[11] we shared strategies for handling the distribution of prior experience. Strategies include: students take an assessment to determine where in the intro sequence they should begin (assessment can be a self- or faculty-administered evaluation); students self-select where to begin based on preference;[20,i] the department offers recitations for true beginners (for which they receive credits); and the department offers a CS0 course (often in Python) for non-majors and students without experience[19,24,32,37,j] and/or implements two, differently paced intro sequences that merge together in CS2 or CS3.[31,k]

2. Implement common assessment across multi-section classes. Having shared assignments, exams,[l] course

e In U.S. universities, schools are either organized on a yearly two-semester basis for four years in which students take 4–5 courses/semester, or on a yearly three-quarter basis for four years in which students take 3–4 courses/quarter. The minimum number of credits for a 4-year degree in the U.S. is 120 credits and at most universities a course counts for 3–4 credits.

f We define computing according to the IPEDS Classification of Instructional Program (CIP) code 11, with CS typically defined as CIP code 11.07.

g Pass is defined by receiving a passing grade and being eligible to enroll in the next course (i.e., if a student must earn a C in CS1 to enroll in CS2, then a C- would be considered fail).

h NCWIT provided many invaluable insights. See https://ncwit.org

i Ensuring the versions are offered at the same time means students choose what is best for their comfort level and not what is best for their class schedule.

j Advisors must understand who should be taking CS0 and advise students accordingly. CS0 must have a pathway to the major and ensure that a student can still complete the major in four years if they start with CS0 in their first or second semester.

k Advising is critical for this approach, as students will automatically opt for the shorter pathway to save on tuition.

l Common assessment does not have to mean a single time slot for exams. Many of the schools the CIC works with successfully create a bank of exam questions that assess the required outcomes, but vary sufficiently as to mitigate cheating. The CIC has not found any significant limitations to this solution.

policies, and grading standards across sections and labs of courses in the intro sequence is key to ensuring that all students can thrive in computing.[8] On the one hand, students without prior coding experience are unlikely to persist if confronted with a "super hard" CS1. On the other hand, a "super easy" CS1 will not prepare students for CS2. Common assessment in CS1–3 is therefore essential for a shared foundation, regardless of prior background and/or class instructor. Also, common assessment allows for a centralizing of resources related to TAs and graders, translating into greater coverage and more timely grading. To implement this change, the department can delegate course management to a rotating course coordinator position. It goes without saying that an added benefit to common assessment is that faculty who teach downstream can be confident as to what was covered.

3. Create assignments that show computing in the context of other fields. By contextualizing the computing curriculum in different domains, students see its applicability to real-world problems and how it connects to other disciplines. This has been extensively investigated by the CS education research community.[6,7,15]

4. Have CS0/CS1 fulfill a general education requirement. This promotes discovery. We acknowledge, however, that this may be infeasible in certain contexts.

5. Right-size degree requirements. Departments must evaluate whether the number of required courses and the prerequisite structure in their CS degree(s) are reasonable.[22] Some departments require 20 or more semester-long computing courses plus math and science requirements. In such cases, it can be impossible for a student to *discover* computing and still graduate in four years. A rule of thumb is that degree requirements can be satisfied in three years—this promotes discovery and mitigates added time to graduation for transfer students. In addition, research has shown that a rigid pre- and co-requisite structure can impact the make-up of computing majors[22] and student retention in the major.[17] Of particular importance is the placement of math prerequisites. We recommend not requiring a math course until the content is *truly* needed.[10] For example, calculus 1 and 2 are not needed for CS1, CS2, or discrete math.[17] This change is key if a university has a high drop/fail/withdraw rate in calculus 1 and/or 2, meaning the class is functionally blocking students from progressing in the CS degree.

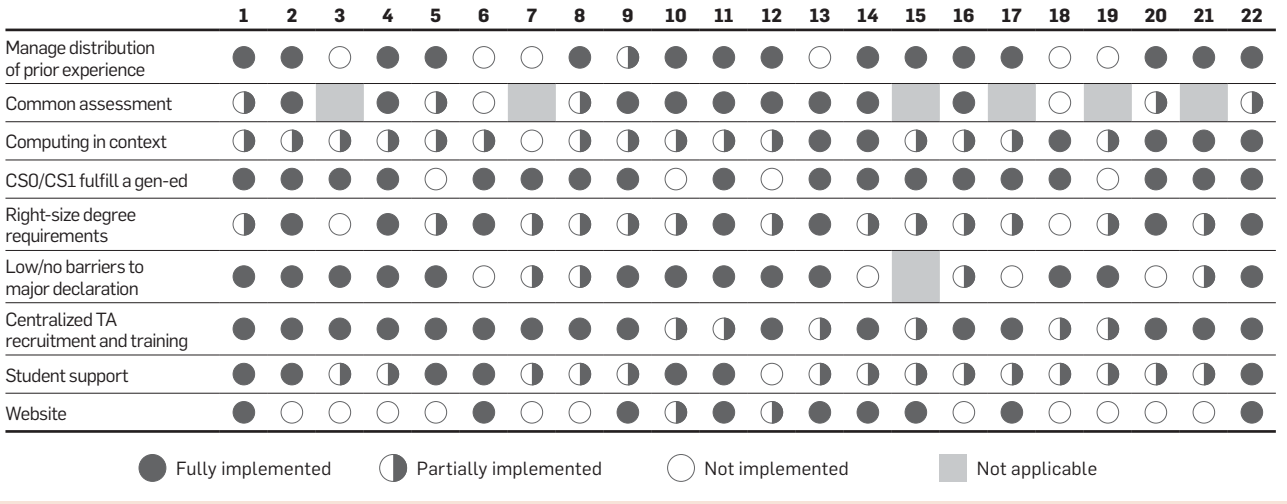
6. Low/no barriers to major declaration. Because of historic booming enrollments in computing,[14,21,m] many schools have struggled to meet student demand. Three common but problematic responses to this problem are: only allowing students to major in computing if they are admitted directly, allowing any student to take the first two to three computing courses and then deciding whether they can be admitted to the major based on their GPA in those classes, and charging more per credit for "high-demand" majors, which is a barrier for low-income students. Limiting enrollment to those admitted directly to the major excludes students from discovering computing at university. Imposing GPA-based enrollment caps favors students who arrive at university with prior coding experience. If a department feels it must cap enrollments, the CIC suggests alternative strategies that can meet the department's needs without sacrificing the ability to attract and retain true beginners (these are discussed in Brodley[12]).

7. Create interdisciplinary computing majors. Often referred to as CS+X in the literature, interdisciplinary computing majors (ICM)—majors built at the intersection of two or more fields of study)—are a powerful strategy for broadening participation in computing.[9,13]

8. Centralize TA recruiting, training, and evaluation. At many universities, the CIC observes little to no training for TAs (and, if there is training, it is often the job of each faculty member to train those TAs assigned to them). Because a student's experience and sense of belonging are often tied to their experience with the TAs, it is vital to train and evaluate TAs (see AIICE[1] and Muzny et al.[28] for publicly available TA training

m As of the writing of this article, undergraduate enrollment in computing majors is cooling at many universities.

Figure 1. Implemented systemic and sustainable changes.



materials). Centralizing recruitment of TAs has the added benefit of removing the biases of individual faculty members and providing a centralized view of which TAs to rehire based on evaluation results.[29]

9. Align co-curricular supports. As schools make changes to their intro sequence and degrees, it is essential to update and align co-curricular supports and arm academic advisors with the information to help students understand and navigate the various pathway options. Supports also include services like tutoring and peer advising, which are scalable and low-cost methods for meeting student need.

10. Present clear degree maps on the department website. Many students will first be exposed to the major via the degree plan on the department's website. As such, it is imperative that the site be both accurate and welcoming. We place a particular emphasis on making sure websites contain multiple sample plans of study, including plans for transfer students, students who enter university without pre-calculus, and students who discover computing in their second year of university.[n]

Some departments require 20 or more semester-long computing courses plus math and science requirements. In such cases, it can be impossible for a student to *discover* computing and still graduate in four years.

Impact

Here, we present results from the 22 universities[o] that have been implementing CIC changes for three or more years. We wait three years because some changes require a year or more to go through university governance and/or be fully implemented. We first summarize the systemic changes made by the schools, then discuss the impact of the changes on who is majoring in CS, sharing data on the enrollment and pass rates in CS1. We conclude by discussing challenges that resulted in some schools being less successful than others.

Changes made across the portfolio. In Figure 1, we list the interventions implemented by each of the 22 universities (per the terms of the CIC's grant agreement, schools are anonymized; they are also not listed here in alphabetical order). In the figure, we indicate the level of implementation for each intervention (see legend). We explain our coding rules below. Note that we do not include interdisciplinary computing majors (ICM) because this intervention was not added until 2024. Also note that in our five-year report[p] we show detailed data for four schools.

Manage distribution of prior experience: Coded as fully implemented if the school follows one of the approaches described in the previous section, otherwise coded as not implemented, with one exception: coded as partially implemented if the school provided extra supports for students new to coding but did not give credit for the additional work.

Common assessment: Coded as fully implemented if the school has implemented common assignments *and* exams in CS1–3. Coded as partially implemented if common assessment is in a subset of CS1–3 and/or if this entails common exams *or* common assignments, but not both. For six schools, this intervention is N/A because the classes have only one section.

Computing in context: Coded as fully implemented if the department has implemented the intervention for all of CS1–3; partially implemented if only for for a subset.

CS0/CS1 fulfill a gen-ed requirement: This is either true (fully implemented) or false (not implemented).

Right-size degree requirements: Coded as fully implemented if three things are true: the number of computing credits is ≤ 60 (including discrete math but excluding other required math); calculus 1 is not a prerequisite to CS1 or CS2; *and* the degree's prerequisite structure is not overly complex.[q]

n See https://cic.northeastern.edu/interventions/undergraduate-interventions/ for the CIC's top 10 tips for websites.

o The 22 universities are: California State University, Long Beach; California State University, Fullerton; Colorado State University; Columbia University/Barnard College; Florida International University; Georgia State University; New Jersey Institute of Technology; Rutgers University; San Jose State University Stony Brook University (SUNY); University of California, Berkeley; University of California, Riverside; University of Central Florida; University of Illinois, Urbana-Champaign; University of Michigan, Ann Arbor; University of Minnesota; University of South Florida; University of Texas, Austin; University of Utah; University of Washington, Seattle; and University of Wisconsin, Madison.

p http://cic.northeastern.edu

q We define reasonable complexity if the curricular complexity $\leq 250^{22}$ (see Slim et al.[36] for a definition of curricular complexity).

Schools are coded as partially implemented if one of the three criteria is met *or* if the school's B.S. does not meet the criteria, but the institution offers a B.A. that does. We note that, across the 22 universities, the B.S. in CS requires on average 56.3 ± 12.3 computing credits and the B.A. in CS requires 40.5 ± 6.5 computing credits.

Low/no barriers to major declaration: Coded as fully implemented if one of the following is true: there are no restrictions to major declaration beyond what is dictated by the host college, or the GPA restriction is ≤ 2.5. Partially implemented means students must be able to apply to the major after entering the university *and* the department either uses a holistic evaluation or has a technical GPA requirement in the range (2.5, 3.2). Otherwise, coded as not implemented.

Centralized TA recruiting and training: Coded as fully implemented if both TA recruiting and training are centralized; partially implemented if only one is in place.

Co-curricular supports: Coded as fully implemented if the school offers CS tutoring, has an advisor caseload of ≤ 375 students, and has a policy requiring that students meet with advisors at key academic milestones. Partially implemented means only one or two of these criteria have been met. If none are met, then coded as not implemented.

Clear degree maps on website: Coded as fully implemented if degree requirements and/or plan of study include pathways for *all* students, including external transfers, those discovering computing in their second year, and those who are not "calc-ready." Coded as partially implemented if there is guidance for some but not all of these pathways. Otherwise, coded as not implemented.

In Figure 1, we see that most schools implemented many but not all interventions, because not all interventions are feasible in all contexts. For example, making CS0/CS1 fulfill a general education requirement or mandating common assessment requires significant faculty buy-in. Other interventions, such as centralizing TA recruiting and training and computing in context, are widely adopted. In many cases, we have provided follow-on funding after the end of the initial grant so schools can implement additional interventions. For example, University 6's leadership did not feel ready to tackle common assessment in the early years of working with us but is now creating a policy to do so. Because at the time of this writing we have only 22 schools that implemented changes three or more years ago, an analysis of *which* interventions have the most impact would not be statistically valid. This is a priority area for future work. Four schools have published results from the work supported by the CIC: Rutgers,[16] University of Washington,[38] University of California, Riverside,[31] and Colorado State University.[17]

Plotting the data for each population, we see that for all populations, the growth in computing majors increased more than the corresponding change in university enrollment.

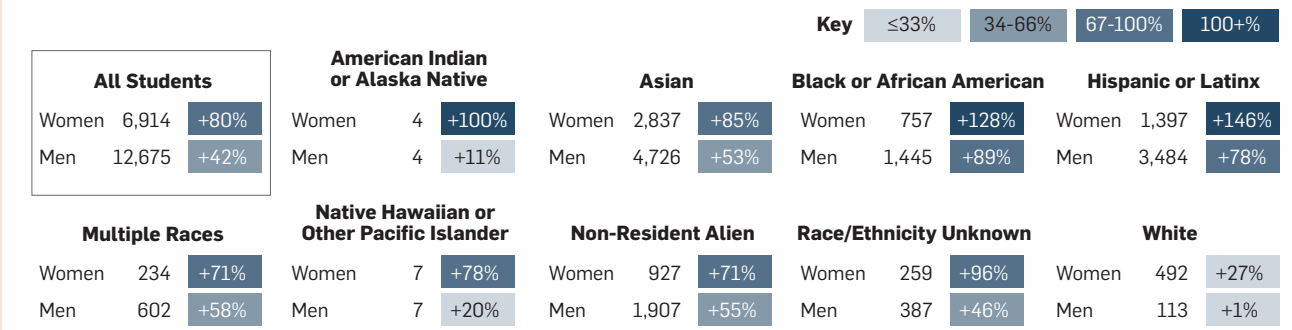
Who is majoring? The ultimate measure of impact for this work is whether the student graduates with the degree. However, graduation data means waiting four or more years from implementation. Thus, for now, our best metric is *who is majoring* in computing. (Note that the timing of major declaration varies by school.) Across the 22 schools, the number of majors grew between 2019 and 2024[r] to 42,558 men and 15,533 women.[s] The change in representation of who is majoring also changed over this period, with the percentage of women majors increasing from 22.4% to 26.7%. Note that results vary across the universities, as discussed in the "Challenges" section below; indeed, the mean increase in women is 4.0 points and the standard deviation is 3.2. In Figure 2, we show the changes for each race/ethnicity and gender grouping in terms of net *new* majors and percentage change (the darker the box, the larger the growth). We see that women majors increased by 80% and 6,914 net-new women majors were added. The most dramatic growth was for Black/African American women and Hispanic women, at 128% and 146%, respectively. The numbers for some groupings are so small (e.g., American Indian/Alaska Native) that the percentage change is not meaningful.

A question that arises is whether

r We compare 2019 to 2024 to make sure that the impact of COVID on pass rates and retention are minimized.

s At the time of finalizing this article, we do not yet have 2025 data, as it takes until late spring for schools to provide their fall data.

Figure 2. Number and percentage change in majors from 2019 to 2024.



this growth is the result of interventions or a function of changes in overall university demographics. In Figure 3, we compare the change in representation in computing (y-axis) and the change in representation in the university (x-axis). We calculate the change in enrollment for a university using IPEDS data.[t] Plotting the data for each population, we see that, for all populations, the growth in computing majors increased more than the corresponding change in university enrollment. Indeed, even populations that are seeing declines in university attendance still show an increase in their representation in computing.

Improvements in pass rates. In Table 1, we show the change in pass rates for CS1 from Fall 2019 to Fall 2024. The pass rates improved for almost

t IPEDS provides overall enrollment in the university but not by major.

Figure 3. Comparison of percentage change in CS majors (y-axis) from 2019 to 2024 vs. percentage change in overall undergraduate university enrollment (x-axis) from 2019 to 2024.

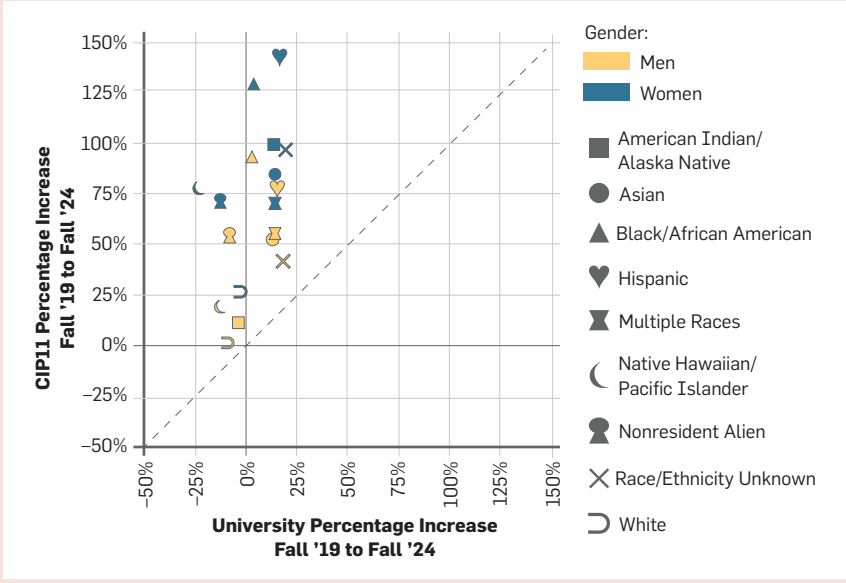


Table 1. Number of students enrolled in CS1 in 2019 and 2024 and pass rates CS1 in 2019 and 2024.

	Women				Men			
	Enrolled (N)		Pass Rate (%)		Enrolled (N)		Pass Rate (%)	
Race/Ethnicity	2019	2024	2019	2024	2019	2024	2019	2024
American Indian or Alaska Native	2	4	100	100	14	17	64	76
Asian	1,346	1,404	89	90	2,763	2,710	87	89
Black or African American	170	332	68	77	456	674	63	75
Hispanic or Latino	446	675	73	75	1,537	1,868	77	80
Multiple Races	119	157	82	84	303	349	80	85
Native Hawaiian or Other Pacific Islander	6	2	67	100	10	13	80	77
Non-Resident Alien	679	744	88	89	1,576	1,574	82	86
Race Unknown	145	206	88	89	233	294	89	87
White	875	760	85	84	3,114	2,316	83	86
All races and ethnicities	3,788	4,284	82	87	9,689	9,701	78	82

every population. We show the results by gender[u] and race/ethnicity. In some cases, the improvements are as high as 8–10 percentage points. The variance among pass rates by race/ethnicity is smaller in 2024 than in 2019, but persists, indicating more still needs to be done.

Threats to the validity of CIC results. Ideally the CIC would compare the results of the 22 schools to a set of similarly sized schools that did not make these changes, but there is no principled way to do this and all that is publicly available via IPEDS is graduation data. This will change in a few years, when we will be able to compare the graduation data of our 22 schools to the graduation data of a random sample from IPEDS. A limitation of the results is that we worked only with schools that graduate 200 or more CS majors a year. Thus, whether the 10 interventions would be effective at smaller schools, and liberal arts colleges in particular, is an open question.

Challenges that impacted some schools' success. The results above show that the portfolio as a whole made significant progress in making CS more equitably accessible. However, progress varies by school, with some schools failing to make progress despite implementing several of the recommended interventions. We have found several reasons for this. The first is a high GPA-based barrier to major declaration. When a high cap is in place, students we have met share that CS1 and CS2 are hypercompetitive and uncollegial. This environment can negatively impact students who might be considering trying CS. In schools with high (>3.5) GPA major declaration thresholds, the demographics changed little. A second confounding factor we observed at two schools was due to university-level budgetary pressure. In both cases, although enrollment in CS continued to grow, strains on university resources led many CS faculty to leave, which reduced the project team's bandwidth to implement the changes.

u While, the CIC's system is set up to collect data for nonbinary students, not every university collects data this way and the numbers are small enough to be identifying, so we omit this data from the table.

The hardest change to make—but perhaps the most powerful—is common assessment. Faculty are often resistant to this change because they feel it infringes on their academic freedom.

The final confounding variable is a lack of positional authority on the project team. Some interventions do not need much in the way of financial support but do demand that the project team have the authority to *require* the changes. Indeed the hardest change to make—but perhaps the most powerful—is common assessment. Faculty are often resistant to this change because they feel it infringes on their academic freedom. This has proven a valuable lesson about the importance of ensuring department leadership is bought into and able to execute on the desired changes. The CIC now requires that the department chair be on the project team.

What Is Next for the CIC?

By 2027, we project a portfolio of 50 or more schools implementing/having implemented these interventions. At that point, we will have sufficient data to more definitively show which interventions and combinations of interventions have the highest efficacy—and in which university contexts. This future research is critical for providing universities a road map for how to chart their systemic change journey once the CIC terminates operations.

In 2024, we launched an effort to understand the challenges and opportunities of interdisciplinary computing majors (ICMs). We are currently working with 10 partner schools to understand the implementation of ICMs in different institutional contexts, and to demonstrate the attractiveness of these degrees to an even broader array of students.

Finally, we are working to identify interventions that ensure equitable access to AI education. We are currently conducting a national landscape study of AI programs (see https://cicmap.ai/) and requirements at the undergraduate level, with the goal of identifying trends and best practices. Between 2025 and 2027, we will issue several RFPs to implement these best practices to ensure that the urgent adoption of AI is an opportunity for all.

Conclusion

Broadening participation in computing has been a goal of the CS community for decades, and the CIC is profoundly grateful to have been

given the opportunity to advance this shared goal by testing the efficacy of interventions *across* universities. The framework we have developed over the past five years has led to systemic, sustainable changes at the implementing universities. Further, having made the agreed upon changes, the universities in this initial portfolio do not require ongoing funding to sustain the impact. Additionally, the CIC has been able to show how to adapt an intervention's implementation based on a university's particular context. The results presented here illustrate that systemic change is both possible and leads to increases in discovery, retention, and declaration of computing majors across all populations. We are excited to add to these results as we grow the number of schools in our portfolio.

Acknowledgments

This project was partially funded by Pivotal and by the National Science Foundation (awards 2216629, 2434248 and 2533723). We would like to thank CIC partner schools, our program managers and technical assistants, and Northeastern University offices of contracting, finance and grants. Christine Alvarado, Valerie Barr, and Manuel Pérez-Quiñones gave us useful feedback on an earlier version of this article.

References

1. AIICE. *AIICE Teaching Assistant Professional Development Course* (2023); https://sites.google.com/mtholyoke.edu/aiiceta-pd-info-site
2. Alvarado, C., Umbelino, G., and Minnes, M. The persistent effect of pre-college computing experience on college CS course grades. *ACM Inroads* 9, 2 (Apr. 2018), 58–64.
3. Babes-Vroman, M. et al. Exploring gender diversity in CS at a large public R1 research university. In *Proceedings of the 2017 ACM SIGCSE Technical Symp. on Computer Science Education*. ACM (2017), 51–56.
4. Barr, V., Brodley, C.E., and Pérez-Quiñones, M. Visualizing progress in broadening participation in computing: The value of context. *Commun. ACM* 67, 7 (Jul. 2024), 46–55.
5. Basnet, R.B. et al. Exploring bimodality in introductory computer science performance distributions. *Eurasia J. of Mathematics, Science and Technology Education* 14, 10 (2018).
6. Bayliss, J.D. and Strout, S. Games as a "flavor" of CS1. In *Proceedings of the 37th SIGCSE Technical Symp. on Computer Science Education*. ACM (2006), 500–504.
7. Boese, E.S., LeBlanc, M.D., and Quinn, B.A. EngageCSEdu: Making interdisciplinary connections to engage students. *ACM Inroads* 8, 2 (May 2017), 33–36.
8. Brodley, C.E. and Gill, C. The BPC relevance of common assessment in the introductory sequence. *Commun. ACM* 67, 7 (2024), 46–55.
9. Brodley, C.E. et al. Broadening participation in computing via ubiquitous combined majors (CS+X). In *Proceedings of the 53rd ACM Technical Symp. on Computer Science Education*. ACM (2022).
10. Brodley, C.E., Quam, M., and Weiss, M. An Analysis of the math Requirements of 199 CS BS/BA segrees at 158 U.S. universities. *Commun. ACM* 67, 8 (2024), 122–131.
11. Brodley, C.E. (2022). Expanding the pipeline: Addressing the distribution of prior experience in CS1. *Computing Research News* 34, 6 (June 2022); http://doi.acm.org/10.1145/1219092.1219093
12. Brodley, C.E. Why universities must resist GPA-based enrollment caps in the case of surging enrollments. *Commun. ACM* 65, 8 (Aug. 2022), 20–22; 10.1145/3544547
13. Brodley, C.E. et al. ACM 2023: CS + X – Challenges and opportunities in developing interdisciplinary computing curricula. *ACM Inroads* 15, 3 (Aug. 2024), 42–50.
14. Camp, T. et al. Generation CS. *ACM Inroads* 8, 2 (May 2017), 44–50.
15. Carter, L. Interdisciplinary computing classes: Worth the effort. In *Proceedings of the 45th ACM Technical Symp. on Computer Science Education*. ACM (2014), 445–450.
16. Centeno, A. P., Hernandez, C., and Pandit, K. Our journey towards a diverse computing program: What worked, where we are, and what we have learned. In *Proceedings of the 56th ACM Technical Symp. on Computer Science Education V. 1*. ACM (2025), 206–212.
17. Ganesan, S., Lionelle, A., Gill, C., and Brodley, C.E. Does reducing curricular complexity impact student success in computer science? In *Proceedings of the 56th ACM Technical Symp. on Computer Science Education V. 1*. ACM (2025), 360–366.
18. Gifford, A. Code.org: Over 10,000 high schools don't teach compsci. *Code.org* (2023); https://www.govtech.com/education/k-12/code-org-over-10-000-high-schools-dont-teach-compsci
19. Haungs, M., Clark, C., Clements, J., and Janzen, D. Improving first-year success and retention through interest-based CS0 courses. In *Proceedings of the 43rd ACM Technical Symp. on Computer Science Education*. ACM (2012), 589–594.
20. Kirkpatrick, M.S. and Mayfield, C. Evaluating an alternative CS1 for students with prior programming experience. In *Proceedings of the 2017 ACM SIGCSE Technical Symp. on Computer Science Education*. ACM (2017), 333–338.
21. Lehman, K.J. et al. Growing enrollments require us to do more: Perspectives on broadening participation during an undergraduate computing enrollment boom. In *Proceedings of the 52nd ACM Technical Symp. on Computer Science Education*. ACM (2021), 809–815.
22. Lionelle, A., Quam, M., Brodley, C.E., and Gill, C. Does curricular complexity in computer science influence the representation of women CS graduates? In *Proceedings of the 55th ACM Technical Symp. on Computing Science Education*. ACM (2024).
23. Margolis, J. *Stuck in the Shallow End: Education, Race, and Computing*. MIT Press (2008).
24. Marling, C. and Juedes, D. CS0 for computer science majors at Ohio University. In *Proceedings of the 47th ACM Technical Symp. on Computing Science Education*. ACM (2016), 138–143.
25. McGill, M.M., Decker, A., and Settle, A. Undergraduate students' perceptions of the impact of pre-college computing activities on choices of major. *Trans. on Computing Education* 16, 4 (2016), Article 15, 33.
26. Morrison, B.B. et al. Evidence for teaching practices that broaden participation for women in computing. In *Proceedings of the 2021 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM (2026) 57–131.
27. Muzny, F., Giordano, M., Stone, E., and Brodley, C.E. Collecting, analyzing, and acting on intersectional, longitudinal data and pass/fail/withdraw rates in computing courses. In *Proceedings of the 55th ACM Technical Symp. on Computing Science Education*. ACM (2024).
28. Muzny, F. and Shah, M.D. Teaching assistant training: An adjustable curriculum for computing disciplines. In *Proceedings of the 54th ACM Technical Symp. on Computing Science Education*. ACM (2023).
29. Muzny, F., Suria, A., and Brodley, C.E. Student application trends for teaching assistant positions. In *Proceedings of the 56th ACM Technical Symp. on Computer Science Education V. 1*. ACM (2025), 798–804.
30. National Center for Education Statistics. The integrated postsecondary education data system (2023); https://nces.ed.gov/ipeds/
31. Pang, A. et al. Creating a second pathway to the computing major. In *Proceedings of the 57th ACM Technical Symp. on Computer Science Education*. ACM (2026).
32. Reed, D. Rethinking CS0 with JavaScript. *SIGCSE Bull.* 33, 1 (Feb. 2001), 100–104.
33. Reboot Representation. Reboot representation report (2018); https://www.rebootrepresentation.org/
34. Rogerson, C. and Scott, E. The fear factor: How it affects students learning to program in a tertiary environment. *J. of Information Technology Education* 9 (2010), 147–171.
35. Sax, L.J., Lehman, K.J., and Blaney, J.M. Expanding the pipeline – Building recruiting and inclusion for diversity (BRAID): Emerging research on diversifying the CS Major. *Computing Research News* 28, 3 (2016); https://cra.org/crn/2016/03/expanding-the-pipeline-building-recruiting-andinclusion-for-diversity-braid-emerging-research-on-diversifying-the-cs-major/
36. Slim, A., Kozlick, J., Heileman, G.L., and Abdallah, C.T. The complexity of university curricula according to course cruciality. *Proc. - 2014 8th Int. Conf. Complex, Intell. Softw. Intensive Syst.* (Oct. 2014), 242–248.
37. Wood, Z.J. et al. Mixed approaches to CS0: Exploring topic and pedagogy variance after six years of CS0. In *Proceedings of the 49th ACM Technical Symp. on Computer Science Education*. ACM (2018), 20–25.
38. Wortzman, B., Chen, M., Natsuhara, M., and O'Rourke, E. Choosing their own way: Guided self-placement for students in an introductory programming sequence. In *Proceedings of the 57th ACM Technical Symp. on Computer Science Education*. ACM (2026).

Carla E. Brodley (c.brodley@northeastern.edu) is a professor at Khoury College of Computer Sciences and Dean of Inclusive Computing at Northeastern University, Boston, MA, USA.

Catherine Gill is the executive director of the Center for Inclusive Computing, Northeastern University, Boston, MA, USA.

Megan Giordano is the director of undergraduate programs at the Center for Inclusive Computing, Northeastern University, Boston, MA, USA.

Caitlin J. Kidder is the director of graduate programs at the Center for Inclusive Computing, Northeastern University, Boston, MA, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3800691

LLMs can dynamically generate error-handling strategies in real time, tailored to specific runtime contexts such as error messages and program states.

BY ZHENSU SUN, HAOTIAN ZHU, BOWEN XU, XIAONING DU, LI LI, AND DAVID LO

Toward Agentic Runtime Healing

RUNTIME ERROR HANDLING[37] is a critical aspect of software reliability engineering. It requires developers to anticipate potential errors and implement corresponding error handlers in advance. Unanticipated runtime errors, which lack corresponding handlers and are naturally uncaught, can lead to severe consequences, such as unexpected program crashes, exposure of sensitive information, and other critical issues.[8] To mitigate such risks, significant efforts have been made to identify potential errors in source code through techniques such as software testing and verification.[19,41] However, identifying all possible runtime errors and providing corresponding exception handlers remains a challenging, often impractical task.

Given the near inevitability of unanticipated runtime errors, minimizing their impact requires dynamic countermeasures. This necessity has driven the development of self-healing systems,[12,28] a longstanding research area focused on enabling software to automatically recover from abnormal states and restore functionality. Importantly, healing a program execution does not involve patching the code to fix an error but rather revising the runtime state, such as variable values and environment configurations, to recover the program's execution. A successful healing process should not only prevent unexpected termination, but also correct the runtime state so the program can continue meaningfully. The core challenge for self-healing systems lies in the fact that specific errors and their locations are unknown in advance, requiring real-time responses. Consequently, these systems rely on predefined heuristic strategies, such as restarting,[4] rolling back to a checkpoint,[5] trying alternative methods,[6] modifying input,[23] reusing existing error handlers,[13]

» key insights

- **Traditional self-healing systems rely on predefined rules (e.g., restart, rollback), which cannot adapt to the diverse range of unanticipated runtime errors. Large language models offer a fundamentally different approach: generating bespoke error-handling code in real time by reasoning over error context.**
- **Without any fine-tuning, GPT-4 can continue execution in 73% of runtime error cases and produce correct results in 39% of them, demonstrating that LLMs already possess substantial capability for runtime error recovery.**
- **The primary barrier to practical adoption is not effectiveness but trustworthiness: 75% of LLM healing behaviors follow recognizable, auditable patterns, suggesting that safety mechanisms like behavior whitelisting can make this approach viable.**

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK.COM



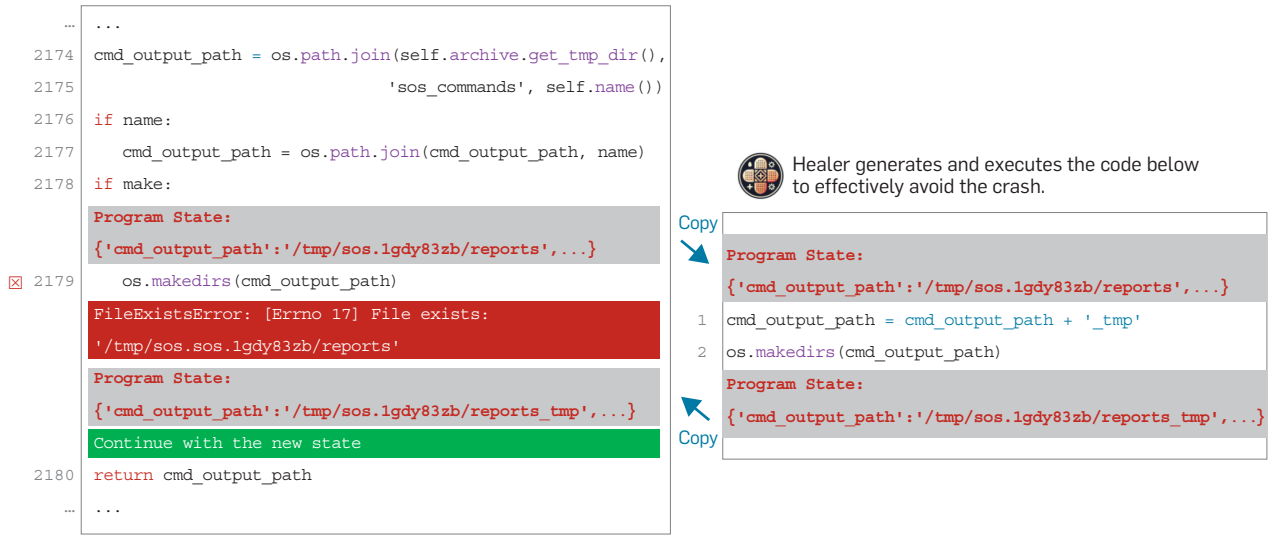
preparing default error handlers,[29] or combining multiple strategies.[7] Despite decades of exploration, the reliance on rule-based strategies limits the adaptability of self-healing systems in handling the diverse range of runtime errors that can occur. The complexity of defining adaptation rules further complicates their implementation in real-world applications. A large-scale industry study highlights this challenge, emphasizing the need for more flexible and adaptive approaches to self-healing.[38]

Recently, the emergence of large language models (LLMs) has introduced new opportunities to revolutionize this field. LLMs have demonstrated exceptional performance in code-related tasks,[14] such as the AlphaCode2 model,[3] which outperformed 85% of human participants in a programming competition. These models possess the ability to understand both source code and natural language, and can generate a piece of source code within seconds. These abilities align well with the requirements of runtime error handling, for example, comprehending runtime error contexts, such as error messages and program states, and generating tailored solutions for unanticipated errors in real time. Specifically, when an unanticipated runtime error occurs, an LLM can be invoked with the runtime error contexts in the prompt and generate an appropriate code script. This script is not merged into the original program but rather is executed dynamically to recover the program from its faulty state. Thus, we propose that using LLMs as runtime error handlers could be a promising pathway toward achieving the longstanding goal of creating truly adaptive and self-healing software systems.

In this article, we take the first step toward this ambitious goal by demonstrating its feasibility. Specifically, we designed a framework, HEALER, for LLM-driven runtime error handling and experimentally evaluated its effectiveness. To illustrate how HEALER facilitates runtime error handling, in Figure 1 we present a motivating example from an open source tool for collecting system logs, SOS[32] (v4.5.5). The extracted code is designed to create a temporary directory in which to save collected logs. However, when executed in a race condition where `os.makedirs()` is called after the directory has already been created, the code crashes at line 2179, resulting in the loss of collected logs. When such unhandled runtime errors occur, HEALER can be invoked to handle the error. It gathers relevant information and instructs its LLM to generate a healing code snippet that seeks to avoid termination caused by the error, enabling the program to continue execution with a corrected state. For this example, the healing code renames the output directory to a new name, avoiding a conflict with the existing directory. Once generated, the healing code is dynamically executed to produce a new program state, allowing the program to continue execution with the updated state.

Following the design of HEALER, we implemented a prototype for Python and evaluated its effectiveness. The evaluation was conducted on four different code datasets, containing a total of 1,185 (code, test case) pairs, where the code raises runtime errors with the test case. We tested HEALER empowered by three state-of-the-art LLMs: two closed-source LLMs, GPT-3.5 and GPT-4, and one open source LLM, CodeQwen. Initially, we investigated whether HEALER, under a zero-shot setting, could effectively avoid interruption and correct faulty program states for runtime errors during execution. The results show that without any fine-tuning, GPT-4 could continue execution in 72.8% of the instances and produce correct results in 39.6% of them. This demonstrates that, even without sophisticated measures, LLMs are capable of correctly handling runtime errors, revealing their great potential for this task. Ad-

Figure 1. A motivating example demonstrating how HEALER handles runtime errors to recover program execution. The code snippet and issue are extracted from SOS,[32] an open source tool for collecting system logs.



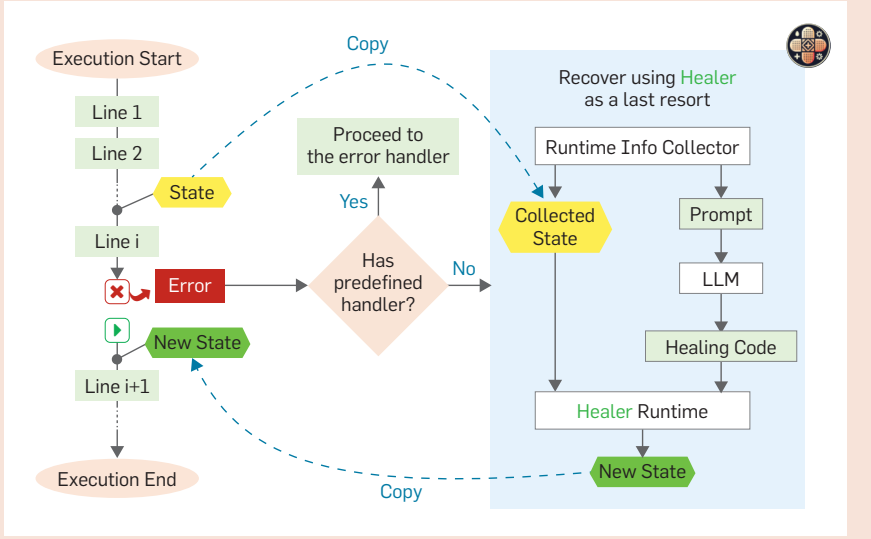
ditionally, we created an instruction-tuning dataset using the (prompt, healing code) pairs generated in our experiments and fine-tuned GPT-3.5 and CodeQwen to assess the effectiveness of fine-tuning. After fine-tuning, the performance of GPT-3.5 and CodeQwen improved by 31.7% and 23.1%, respectively, in instances free of interruption, and by 26.8% and 65.8% in instances yielding correct results. Notably, the fine-tuned GPT-3.5 achieved performance comparable to that of GPT-4. We thus conclude that using LLMs for runtime error handling is a promising direction with great potential.

Although HEALER has shown promising results, there are remaining challenges that must be addressed before LLMs can be applied in practice. The foremost challenge is the trustworthiness of LLM-generated code, as the generated code is executed directly, and LLMs may produce incorrect results, introducing potential risks to the program. To better understand this trustworthiness issue, we empirically investigated the healing code generated by GPT-4 and observed some risky behaviors, such as ignoring key logic in the original code, which could lead to unexpected program behaviors. Therefore, how to mitigate potential risks from the black box of LLMs is a challenging and important problem. Fortunately, there are many promising solutions. For example, we can predefine a set of "safe" functions, such as a function for data saving, and only allow the LLM to use these functions to perform the healing. With such measures, we may not be far from achieving a fully self-healing system.

Proposed Framework

HEALER handles errors relying on an integrated LLM, which generates a piece of code, named *healing code*, by understanding the error context. The integrated LLM refers to a model that is logically incorporated into the automated runtime error-handling loop, regardless of its physical deployment. It enables the use of diverse back-end models (either local or remote) based on computing resource availability. The workflow of HEALER is illustrated in Figure 2. To ease understanding, we assume the framework is being applied to an interpreted programming language, such as Python or JavaScript, where the code is executed line by line, and the runtime environment automatically persists program state between each execution step. Formally, consider a program P with n lines, represented as $P = \{L_1, L_2, ..., L_n\}$. If an error E occurs at line L_i without an available error handler, HEALER is activated. HEALER first collects the error context, including the error message, the program state S, the entire code snippet, and the erroneous line, to construct a prompt. The integrated LLM is then instructed with this prompt to generate a piece of healing code $L_i^{'}$. This code is executed under the state S of the original program. If the execution of $L_i^{'}$ is successful, the resulting program state $S^{'}$ is updated back to the original program state, allowing the program to continue from line L_{i+1}. Notably, while the state is corrected for the current session, the original program is not permanently fixed and the bug will still need to be reproduced and fixed in future development. If the execution of the healing code still raises errors, the program is terminated and the error is reported, following traditional error-handling mechanisms.

Figure 2. The HEALER workflow. When a runtime error occurs and no predefined error handler is available, HEALER is activated. It collects the error context, prompts the LLM, and generates healing code. The healing code is then executed in HEALER's runtime environment to produce a new program state, allowing the program to continue execution.



The HEALER framework consists of three main components:

▸ *Runtime information collector.* Once HEALER is activated, the runtime information collector gathers necessary information from the runtime environment. Specifically, HEALER collects four types of information: the source code, error message, error location, and program state before the error is raised. This information is essential for a human developer to understand the error and write the healing code; thus, HEALER collects it to prompt the LLM effectively.

▸ *Large language model.* As illustrated in Figure 3, HEALER interacts with the LLM using a structured prompt template that combines task instructions with relevant context fields. The instructions outline the task description, goals, and constraints, supplemented by an example to guide the LLM's generation. Following this, the template is populated with the collected error context; specifically, the entire source code, error message, and program state are inserted into their respective fields. Adhering to OpenAI's best practices for prompt engineering,[25] each field is clearly demarcated by a section title, including "Error Message," "Program State," and "Buggy Code." Notably, the error location is not designated as a standalone section. Instead, we highlight the error-raising logic by wrapping the error-raising block within

<error> and </error> tags directly in the source code. Due to space limitations, the full prompt is provided in our artifacts.[a]

▸ *Healing runtime.* The healing code generated by the LLM needs to be executed to produce a new program state. We refer to the runtime that executes the healing code as the *healing runtime*. A straightforward approach is to execute the healing code in the same runtime as the original program. However, to mitigate potential security risks associated with LLM-generated code, alternative runtime environments can be used. For example, the healing runtime could be a new session, a sandbox, or a virtual machine, depending on the required security level. Notably, the execution does not re-run the original program from the beginning but rather executes the healing code with the program state that existed before the error occurred.

a https://github.com/v587su/Healer

Evaluation of Healer

We further evaluated the effectiveness of HEALER in handling runtime errors. To conduct this evaluation, we implemented a proof-of-concept Python code executor that accepts

Figure 3. An example of the prompt construction workflow in HEALER. The prompt is constructed based on a predefined template.

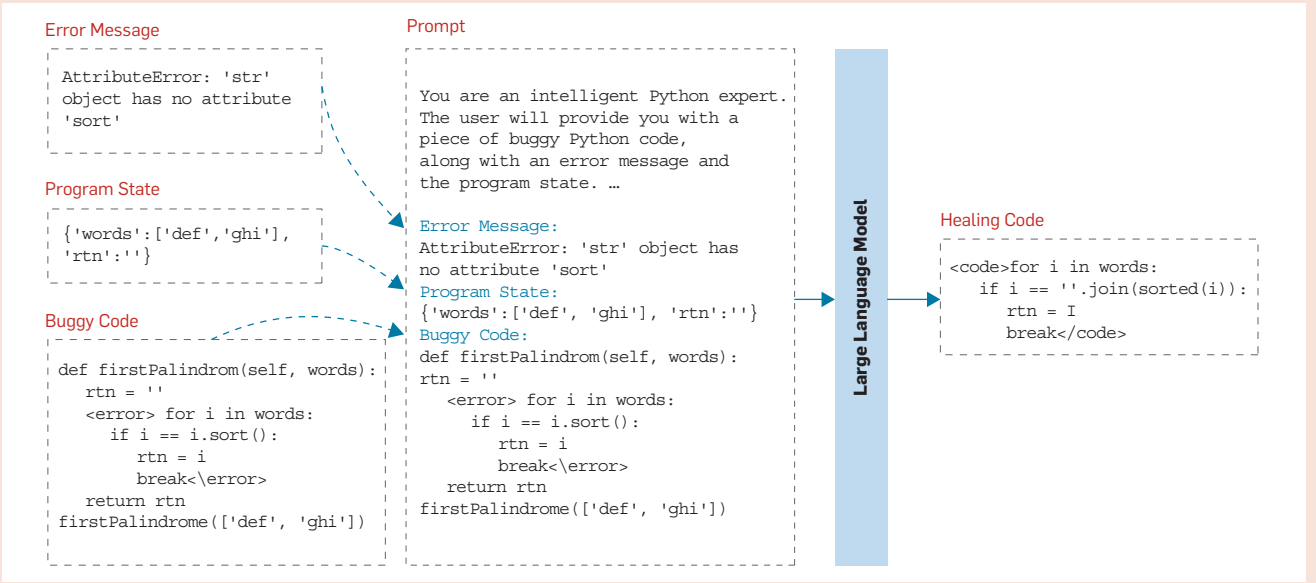


Table 1. The performance of GPT-4, GPT-3.5, and CodeQwen on all of the benchmarks in a zero-shot setting. Benchmarks with model names in parentheses were generated by that specific LLM. CORR., PROC. and FAIL. respectively refer to the percentage of healing results where the program produces correct results, proceeds with execution, or still raises errors.

Benchmark	#Errors	GPT-4			GPT-3.5			CodeQwen		
		CORR.	PROC.	FAIL.	CORR.	PROC.	FAIL.	CORR.	PROC.	FAIL.
CodeNet	228	42.1%	73.2%	26.8%	31.1%	52.6%	47.4%	16.7%	39.9%	60.1%
HumanEval (CodeLlama)	70	47.1%	62.9%	37.1%	35.7%	57.1%	42.9%	12.9%	32.9%	67.1%
HumanEval (CodeT5+)	57	36.8%	64.9%	35.1%	21.1%	50.9%	49.1%	14.0%	36.8%	63.2%
HumanEval (GPT-3.5)	43	23.3%	72.1%	27.9%	7.0%	37.2%	62.8%	14.0%	46.5%	53.5%
HumanEval (GPT-4)	19	57.9%	100.0%	0.0%	21.1%	63.2%	36.8%	10.5%	21.1%	78.9%
HumanEval (Mistral)	37	35.1%	64.9%	35.1%	18.9%	45.9%	54.1%	10.8%	27.0%	73.0%
HumanEval (StarCoder)	69	37.7%	58.0%	42.0%	30.4%	39.1%	60.9%	18.8%	34.8%	65.2%
MBPP (CodeLlama)	30	40.0%	80.0%	20.0%	26.7%	50.0%	50.0%	13.3%	30.0%	70.0%
MBPP (CodeT5+)	49	44.9%	75.5%	24.5%	16.3%	38.8%	61.2%	10.2%	40.8%	59.2%
MBPP (GPT-3.5)	14	7.1%	92.9%	7.1%	0.0%	50.0%	50.0%	0.0%	28.6%	71.4%
MBPP (GPT-4)	3	33.3%	100.0%	0.0%	33.3%	33.3%	66.7%	0.0%	0.0%	100.0%
MBPP (Mistral)	73	34.2%	68.5%	31.5%	13.7%	34.2%	65.8%	11.0%	34.2%	65.8%
MBPP (StarCoder)	57	36.8%	63.2%	36.8%	29.8%	49.1%	50.9%	12.3%	42.1%	57.9%
LiveCodeBench	47	17.0%	76.6%	23.4%	10.6%	51.0%	49.0%	6.4%	29.8%	70.2%
DebugBench	436	40.6%	77.5%	22.5%	22.2%	49.3%	50.7%	12.2%	33.0%	67.0%

source code strings as input and executes them using the HEALER process.

LLMs can effectively handle runtime errors. We evaluated HEALER on four benchmarks: CodeNet[16] for human-written code; HumanEval+, MBPP+,[22] and LiveCodeBench[17] for LLM-generated code; and DebugBench[34] for code with implanted bugs. For CodeNet and DebugBench, we executed the code samples with the provided test cases and retained the (code, test case) pairs that raised runtime errors, yielding 228 and 436 instances, respectively. In the documentation for HumanEval+ and MBPP+, multiple LLMs are evaluated, and the code snippets generated during the evaluation are publicly available.[10] We adopted the code snippets generated by six different LLMs, including GPT-3.5 (ChatGPT),[24] GPT-4 (gpt-4-1106-preview),[1] CodeLlama (34B),[30] CodeT5+ (16B),[36] Mistral (7B),[18] and StarCoder (15B),[20] then followed the same execution-then-selection process to obtain the instances. Finally, we obtained 295 instances from HumanEval+ and 226 instances from MBPP+. Similarly, we collected the error-raising code snippets generated by GPT-4 (gpt-4-1106-preview) from LiveCodeBench as a control group that was free from any potential data-leakage issues, resulting in 47 samples. For each instance, we executed its code snippet with the paired test case and handled the runtime errors using the three LLMs, GPT-3.5, GPT-4, and CodeQwen, following the framework described in Figure 1. We recorded the final execution status of each instance and computed the three metrics: CORRECT, PROCEED, and FAIL, respectively indicating whether the program finally produced correct results expected by the test case, whether the program could continue its execution, and whether the healing code still raised errors.

The results, shown in Table 1, demonstrate that HEALER can effectively heal runtime errors, thereby improving program availability. On average, across all benchmarks, GPT-4, GPT-3.5, and CodeQwen can proceed with 73.0%, 48.3%, and 35.1% of the executions, respectively, and help 38.7%, 23.5%, and 13.0% of the programs produce correct results. This provides compelling evidence that using LLMs to handle runtime errors is a feasible and promising direction. Among the LLMs, GPT-4 consistently achieves the best performance across all benchmarks, with the highest CORRECT and PROCEED metrics. For example, on the DebugBench benchmark, GPT-4 successfully completed the execution of 338 instances and corrected the faulty states for at least 177 instances, accounting for 77.5% and 40.6% of the total errors. Even for code snippets generated by LLMs, such as those in HumanEval, MBPP, and LiveCodeBench, HEALER improved the execution outcomes. For instance, GPT-4 enhanced the execution results of the code it generated by successfully correcting 11 additional instances in HumanEval and one additional instance in MBPP. Moreover, HEALER achieved comparable performance on LiveCodeBench, demonstrating the negligible impact of potential data-leakage issues. This illustrates that LLMs can be used to mitigate the runtime errors caused in their own generated code.

While we primarily focus on comparing different LLMs, it is worth contextualizing these results against traditional self-healing strategies, such as restarting, retrying, and steamrolling (i.e., logging the runtime error and continuing execution despite it). In the context of our benchmarks, the runtime errors are deterministic errors (e.g., IndexError, AttributeError, NameError) instead of environmental failures. Consequently, traditional strategies like retry or rollback would yield a 0% success rate in proceeding execution, as re-executing the same erroneous logic without adaptive healing would lead to the same crash. Similarly, an error steamroller[2,9] might allow the program to survive, thereby improving the PROCEED metric. However, it cannot correct the program state to complete the task, resulting in an extremely low CORRECT rate. For example, when we adopted an error steamroller that exceeded the code execution anyway, the CORRECT metric was only 4.3%. This underscores HEALER's unique value in bridging the gap between simple crash avoidance and functional recovery.

Takeaway 1: HEALER can effectively handle runtime errors and improve the availability and reliability of programs. Notably, GPT-4-based HEALER can continue execution in 73.0% of the instances and produce correct results in 38.7% of the instances.

Fine-tuning further improves HEALER's effectiveness. Using the data points that produced correct results in the previous experiment, we created an instruction-tuning dataset to fine-tune the LLMs for runtime error handling. This dataset consisted of 586 (prompt, healing code) pairs, generated from samples across all benchmarks except the human-written benchmark, CodeNet, which is reserved as the test set for the fine-tuned LLMs. Specifically, we ran GPT-3.5 and GPT-4 to handle the runtime errors in each instance from the selected benchmarks and recorded the healing code. If the healing result was CORRECT, we paired the prompt with its corresponding healing code as a sample in the dataset. With this dataset, we fine-tuned two LLMs, GPT-3.5 and CodeQwen, while using GPT-4's performance as a reference. Specifically, GPT-3.5 was fine-tuned via the commercial API provided by OpenAI, where the hyperparameters follow its default settings. CodeQwen was

Table 2. The comparison between fine-tuned and original versions for GPT-3.5 and CodeQwen. GPT-4 is used as a reference, as we do not have access to its fine-tuning API.

LLM	Version	CodeNet (228)	
		CORRECT	PROCEED
GPT-4	Original	96	167
GPT-3.5	Original	71	120
	Fine-tuned	**90** | 26.8%↑	**158** | 31.7%↑
CodeQwen	Original	38	91
	Fine-tuned	**63** | 65.8%↑	**112** | 23.1%↑

fine-tuned on our machine using low-rank adaptation (LoRA), a parameter-efficient fine-tuning technique,[15] with its hyperparameters available in our artifacts.[a] After fine-tuning, we evaluated the fine-tuned LLMs on CodeNet.

The results, presented in Table 2, show that fine-tuning effectively enhances the LLMs' performance in handling runtime errors. Both GPT-3.5 and CodeQwen exhibit substantial improvements, with GPT-3.5 achieving 26.8% more CORRECT instances and 31.7% more PROCEED instances, and CodeQwen achieving 65.8% more CORRECT instances and 23.1% more PROCEED instances. Remarkably, the fine-tuned GPT-3.5 successfully produces 90 CORRECT instances, closely approaching GPT-4's performance of 96 CORRECT instances. Similarly, the fine-tuned CodeQwen also achieves competitive results comparable to GPT-3.5. This indicates that fine-tuning can be a powerful approach to developing task-specific LLMs for runtime error handling, potentially surpassing general-purpose LLMs in effectiveness. Therefore, a domain-specific LLM for runtime error handling is a promising direction.

Takeaway 2: Fine-tuning significantly improves LLM performance in handling runtime errors. GPT-3.5 demonstrates a 26.8% increase in CORRECT instances and a 31.7% increase in PROCEED instances on CodeNet, achieving performance comparable to GPT-4.

Remaining Challenges

HEALER operates autonomously, executing code generated by LLMs to handle runtime errors without human intervention. However, research has shown that LLM-generated code can exhibit unintended behaviors, introducing risks when executed.[27,39] This issue has also been highlighted in other studies that execute model-generated code directly, such as LExecutor,[33] raising concerns about the reliability of HEALER. For instance, one might ask, "What if the LLM deletes my database?"

LLMs can introduce unexpected behaviors. To better understand the risks, we collected all the incorrect healing code (i.e., the healing code that successfully proceeds the execution but finally produces incorrect results) generated by GPT-4 on the DebugBench. This process yielded 204 pieces of healing code. Healing code that failed to execute in the HEALER runtime was excluded from the analysis, since it did not affect the original runtime environment. Two authors, each with more than three years of experience as Python developers, manually examined these collected instances to identify the behaviors that LLMs might introduce when healing runtime errors. The goal was to empirically discover the behaviors an LLM could cause during the healing process. Specifically, the two authors reviewed each healing code snippet together and discussed the behaviors introduced compared to the original error-raising code. The identified behaviors were incrementally added to a list. Note that we did not aim to create a comprehensive taxonomy of all possible behaviors but rather to provide a quantitative demonstration of HEALER's behaviors.

Fine-tuning can be a powerful approach to developing task-specific LLMs for runtime error handling, potentially surpassing general-purpose LLMs in effectiveness.

In total, we identified 19 distinct behaviors, detailed in Table 3. Some of these behaviors, such as omitting part of the context (2.4%) or omitting return statements (2.9%), are risky and can disrupt the expected flow of data, leading to unexpected execution paths. Therefore, without further constraints, the recovery performed by HEALER is not guaranteed to be correct or safe. Nonetheless, our findings indicate that these risks are manageable because the majority of these behaviors follow distinct patterns that can be precisely identified through rule-based comparisons between the healing code and the error-raising code. Specifically, 75% of the investigated samples exhibit recognizable behaviors, such as changing variable values (21.0%), initializing undefined variables (10.2%), adjusting types (5.9%), or importing new libraries (27.3%). These behaviors can be identified through static checks based on a predefined whitelist. In this way, HEALER can be conservatively configured to perform only behaviors that are not risky to the guarded program. While these restrictions may limit HEALER's capabilities, they significantly reduce the uncertainty associated with its operation.

A trustworthy future? Humans also make mistakes. However, with techniques such as software testing, we can identify and mitigate these risks to an acceptable level. Similarly, in addition to developing more powerful LLMs, supporting mechanisms for HEALER are crucial for ensuring reliability. As a newly proposed idea, the ecosystem for HEALER is currently underdeveloped. Below, we discuss potential methods to help LLMs behave more reliably. By implementing these strategies, we can move closer to a future where HEALER operates reliably and safely within software systems, minimizing risks while maximizing its potential for automated error recovery.

Safety checks. Behaviors such as context omission (skipping parts of the original logic) or return omission (omitting a return statement) can significantly disrupt the expected flow of a program. Mechanisms like static checkers can be used to prevent the execution of potentially harmful code. For example, we can define a whitelist of behaviors that HEALER is allowed to perform and use static checkers to ensure the healing code does not contain any behaviors outside this whitelist. Additionally, techniques used for source code, such as dynamic testing, static analysis, and verification, may also be applicable to the healing code.

HEALER-aware programming. Traditional software systems are fundamentally deterministic, and the uncertainty introduced by HEALER can sometimes be unacceptable. To better accommodate HEALER, software systems need to be designed with its potential interventions in mind. For example, since HEALER may change variable values, the system should include additional mechanisms to handle these changes, such as warning users to be cautious about the system's outputs or maintaining detailed error logs for future bug fixing. Moreover, it is important to establish a "comfort zone" for HEALER within the system, where its interventions are acceptable and unlikely to cause significant harm. This comfort zone could encompass non-critical modules or components, such as user interface elements, where the potential side effects of HEALER's operations are more tolerable. By confining HEALER's activities to these safe areas, developers can benefit from automated error recovery without exposing more sensitive parts of the system to unnecessary risk.

Reliable skillsets for LLMs. Inspired by practices in AI agents for software development,[35] a possible solution is to provide a toolkit as a skill set for the LLM. The LLM inside HEALER would then be restricted to using only the functions in this toolkit. For example, the toolkit could provide a predefined data-saving function, and the LLM could use this function to save data that might otherwise be lost. Since these functions are predefined, the behaviors of HEALER become predictable, thereby mitigating risks.

Discussion

Here, we discuss some potential issues with HEALER, including its deployment implications and extension to other programming languages.

Infrastructure and deployment implications. Technically, HEALER requires access to runtime state information (e.g., stack traces and variable states), an execution environment (e.g., a sandbox, container, or lightweight virtual machine) to properly execute LLM-generated healing code, and access to an LLM service, either via a remote API or a locally deployed model. Among these prerequisites, the LLM service imposes additional requirements on the software, where the choice between remote APIs and local deployment involves distinct architectural trade-offs. Using remote APIs requires stable network connectivity to the server-hosted model. Conversely, employing a local model requires the hosting device to possess sufficient computational power to support inference. To address this, various techniques, such as model compression[31] and quantization,[21] have been developed to enable inference on user devices. For example, LLAMA.cpp[11] can efficiently run gpt-oss-20B[26] on a single M3 Ultra MacBook. As the field advances, we anticipate further reductions in the resource requirements for LLMs.

HEALER in real-world systems. To effectively apply HEALER in real-world applications, some engineering challenges need to be addressed. For in-

Table 3. A summary of the behaviors observed in the healing code generated by GPT-4.

Recognizable behaviors (consistent patterns identifiable by static-analysis tools)	Count
No Action: Repeat the code without any semantical changes	1 (0.5%)
Value Change: Assign/define a new value for a variable/constant	43 (21.0%)
Value Change (IF): Conditionally assign a new value for a variable	13 (6.3%)
Variable Initialization: Initialize an undefined variable	21 (10.2%)
Type Change: Set/modify the type of a variable/constant	12 (5.9%)
Library Import: Import a new library/API	56 (27.3%)
Function Invocation: Invoke an existing custom function with new arguments	3 (1.5%)
Variable Change: Change the reference to a different variable	20 (9.8%)
Operator Change: Use a different operator in a statement or expression	7 (3.4%)
Unrecognizable behaviors (irregular or unpredictable patterns)	**Count**
Function Definition: Define a new function and utilize it in the logic	1 (0.5%)
Function Replacement: Implement the logic as an alternative to a function call	8 (3.9%)
Type Change and Usage: Set a new type to a variable and utilize it in the logic	2 (1.0%)
API Replacement: Invoke an alternative API	12 (5.9%)
Indexing Omission: Omit the indexing to a variable	1 (0.5%)
Library Import and Usage: Import a new library and utilize it in the logic	11 (5.4%)
Context Omission: Skip the remaining logic in error-raising code, except for specific lines	5 (2.4%)
Return Omission: Omit a return statement	6 (2.9%)
Context Duplication: Repeat the entire given code context	1 (0.5%)
Function Call Omission: Omit the invocation of an undefined function	3 (1.5%

stance, real-world software systems can contain millions of lines of code, the majority of which are irrelevant to any single runtime error. Consequently, for LLMs, reading through the entire codebase for every error is inefficient. This necessitates a targeted context retrieval strategy to identify code segments relevant to the error trajectory, methods that are already widely studied in repository-level tasks such as bug localization[41] and repository code completion.[40] Furthermore, HEALER must integrate seamlessly with existing error-recovery mechanisms. For example, it should function alongside traditional rollback strategies (e.g., snapshot restoration) so that if HEALER fails to resolve the error, the system can default to these predefined backup measures.

Extension to other programming languages. While our current experiments implement HEALER in Python, the underlying framework for handling runtime errors is designed to be language-agnostic. Interpreted languages (e.g., Python, JavaScript, Ruby) naturally fit HEALER because they natively support dynamic code execution. In contrast, compiled languages (e.g., Java, C++, Rust) require a compilation step prior to execution, introducing architectural complexity to the dynamic execution process. However, this barrier is not insurmountable through specific language features. For instance, Java provides the Java Compiler API to compile code fragments at runtime, and C++ allows for dynamic loading of shared libraries. While some engineering effort is required to achieve this, extending HEALER to these languages is promising. We look forward to exploring these extended implementations in future work.

Conclusion

AI-driven runtime error handling represents a promising approach to enhancing the reliability of software systems and could serve as a new pathway toward achieving self-healing systems. For the first time, we propose this idea and provide a proof-of-concept implementation to demonstrate its feasibility. Through our implementation, we demonstrate that HEALER can handle runtime errors with a high success rate. However, significant challenges remain, particularly regarding the uncertainty of LLM behavior and the compatibility of such systems with existing software architectures. We hope this work will inspire further research in this area and pave the way for the future of AI-driven runtime error handling, ultimately leading to more robust and resilient software systems. C

References

1. Achiam, J. et al. GPT-4 technical report. arXiv preprint (2023), arXiv:2303.08774.
2. ajalt. fuckitpy: The Python error steamroller (2026); https://github.com/ajalt/fuckitpy
3. AlphaCode Team, Google DeepMind. AlphaCode 2 technical report (2023); https://storage.googleapis.com/deepmind-media/AlphaCode2/AlphaCode2_Tech_Report.pdf
4. Candea, G. et al. JAGR: An autonomous self-recovering application server. In *2003 Autonomic Computing Workshop*. IEEE (2003), 168–177.
5. Carzaniga, A. et al. Automatic recovery from runtime failures. In *2013 35th Intern. Conf. on Software Engineering*. IEEE (2013), 782–791.
6. Carzaniga, A., Gorla, A., Perino, N., and Pezzè, M. Automatic workarounds for web applications. In *Proceedings of the Eighteenth ACM SIGSOFT Intern. Symp. on Foundations of Software Engineering*. ACM (2010), 237–246.
7. Chang, H., Mariani, L., and Pezze, M. Exception handlers for healing component-based systems. *ACM Trans. on Software Engineering and Methodology* 22, 4 (2013), 1–40.
8. Common Weakness Enumeration. CWE-248: Uncaught exception (4.14); https://cwe.mitre.org/data/definitions/248.html
9. Diamond, M. fuckitjs: The original Javascript error steamroller (2026); https://github.com/mattdiamond/fuckitjs
10. EvalPlus. EvalPlus Releases (2024); https://github.com/evalplus/evalplus/releases
11. Gerganov, G. llama.cpp: Port of Facebook's LLaMA model in C/C++ (2023); https://github.com/ggerganov/llama.cpp
12. Ghosh, D., Sharman, R., Rao, H.R., and Upadhyaya, S. Self-healing systems—survey and synthesis. *Decision Support Systems* 42, 4 (2007), 2164–2185.
13. Gu, T. et al. Automatic runtime recovery via error handler synthesis. In *Proceedings of the 31st IEEE/ACM Intern. Conf. on Automated Software Engineering*. ACM (2016), 684–695.
14. Hou, X. et al. Large language models for software engineering: A systematic literature review. arXiv preprint (2023), arXiv:2308.10620.
15. Hu, E.J. et al. Lora: Low-rank adaptation of large language models. arXiv preprint (2021), arXiv:2106.09685.
16. IBM. Project CodeNet (2024); https://github.com/IBM/Project_CodeNet
17. Jain, N. et al. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *The Thirteenth Intern. Conf. On Learning Representations* (2025).
18. Jiang, A.Q. et al. Mistral 7B. arXiv preprint (2023),
19. Kochhar, P.S. et al. Understanding the test automation culture of app developers. In *2015 IEEE 8th Intern. Conf. on Software Testing, Verification and Validation*. IEEE (2015), 1–10.
20. Li, R. et al. StarCoder: May the source be with you! arXiv preprint (2023) arXiv:2305.06161.
21. Liang, T. et al. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* 461 (2021), 370–403.
22. Liu, J., Xia, C.S., Wang, Y., and Zhang, L. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems* 36 (2024).
23. Long, F. et al. Automatic input rectification. In *2012 34th Intern. Conf. on Software Engineering*. IEEE (2012), 80–90.
24. OpenAI. GPT-3.5 Turbo model documentation (2024); https://platform.openai.com/docs/models/gpt-3-5-turbo
25. OpenAI. Prompt engineering: Use delimiters to clearly indicate distinct parts of the input (2024); https://platform.openai.com/docs/guides/prompt-engineering/tactic-use-delimiters-to-clearly-indicate-distinct-parts-of-the-input
26. OpenAI. openai/gpt-oss-20b - Hugging Face (2025); https://huggingface.co/openai/gpt-oss-20b
27. Pearce, H. et al. Asleep at the keyboard? Assessing the security of github copilot's code contributions. In *2022 IEEE Symp. On Security and Privacy*. IEEE (2022), 754–768.
28. Psaier, H. and Dustdar, S. A survey on self-healing systems: approaches and systems. *Computing* 91 (2011), 43–73.
29. Rinard, M.C. et al. Enhancing server availability and security through failure-oblivious computing. In *Proceedings of the 6th conf. on Symp. on Operating Systems Design & Implementation - Volume 6*. USENIX (2004), 21–21.
30. Roziere, B. et al. Code Llama: Open foundation models for code. arXiv preprint (2023), arXiv:2308.12950.
31. Shi, J. et al. Compressing pre-trained models of code into 3 mb. In *Proceedings of the 37th IEEE/ACM Intern. Conf. on Automated Software Engineering*. IEEE (2022), 1–12.
32. sosreport. sosreport/sos: A unified tool for collecting system logs and other debug information (2024); https://github.com/sos-project/sos
33. Souza, B. and Pradel, M. LExecutor: Learning-guided execution. In *Proceedings of the 31st ACM Joint European Software Engineering Conf. and Symp. on the Foundations of Software Engineering*. ACM (2023), 1522–1534.
34. Tian, R. et al. Debugbench: Evaluating debugging capability of large language models. arXiv preprint (2024), arXiv:2401.04621.
35. Wang, X. et al. Openhands: An open platform for AI software developers as generalist agents. arXiv preprint (2024), arXiv:2407.16741.
36. Wang, Y. et al. CodeT5+: Open code large language models for code understanding and generation. arXiv preprint (2023), arXiv:2305.07922.
37. Weimer, W. and Necula, G.C. Finding and preventing run-time error handling mistakes. In *Proceedings of the 19th annual ACM SIGPLAN Conf. on Object-oriented programming, systems, languages, and applications*. ACM (2004), 419–431.
38. Weyns, D. et al. Preliminary results of a survey on the use of self-adaptation in industry. In *Proceedings of the 17th Symp. on Software Engineering for Adaptive and Self-Managing Systems*. ACM (2022), 70–76.
39. Yang, Z. et al. Robustness, security, privacy, explainability, efficiency, and usability of large language models for code. arXiv preprint (2024), arXiv:2403.07506.
40. Zhang, F. et al. RepoCoder: Repository-level code completion through iterative retrieval and generation. In *Proceedings of the 2023 Conf. On Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, (Eds.). Association for Computational Linguistics (2023), 2471–2484.
41. Zhou, J., Zhang, H., and Lo, D. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports. In *2012 34th Intern. Conf. on Software Engineering*. IEEE (2012), 14–24.

Zhensu Sun is a Ph.D. student at Singapore Management University, Singapore, Singapore.

Haotian Zhu is a research engineer at Singapore Management University, Singapore, Singapore.

Bowen Xu is a assistant professor at North Carolina State University, Raleigh, NC, USA.

Xiaoning Du (xiaoning.du@monash.edu) is a senior lecturer at Monash University, Melbourne, Victoria, Australia.

Li Li is a professor at Beihang University, Beijing, China.

David Lo is a professor at Singapore Management University, Singapore, Singapore.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

research highlights

P. 98
Technical Perspective
NVMR for Batteryless Computing
By Brandon Lucia

P. 99
Non-Volatile Memory Renaming for Safe Batteryless Computing
By Abhishek Bhattacharyya, Abhijith Somashekhar, and Joshua San Miguel

DOI:10.1145/3843283

To view the accompanying paper, visit doi.acm.org/10.1145/3840403 rh

Technical Perspective
NVMR for Batteryless Computing

By Brandon Lucia

IT HAS BEEN 10 years since we first wrote about intermittent computing, and research in this area has come a long way to make *batteryless* devices more capable. The accompanying paper on the NVMR architecture opens the door to more capable, correct batteryless systems. Intermittent computing has not always been in the research zeitgeist, so here is an overview. An intermittent computing device is a batteryless hardware/software system that harvests energy from its environment into a tiny capacitor. The system computes until energy is exhausted, when the system abruptly stops to collect more energy. After collecting enough energy to do more work, the system turns back on and the cycle continues. A system can operate indefinitely harvesting solar, RF, or mechanical energy for a range of use cases, such as environmental sensing, wearables, and smart buildings.

Intermittent computing devices operate without batteries, allowing deployment to environments where batteries are infeasible: inhospitable environments like space and glaciers, and operationally infeasible environments with unacceptable costs for battery production, maintenance, and disposal. The abundant uses for these systems are an exciting vision, but it isn't free.

Intermittent operation causes problems for programmers. Programming languages and developers assume *continuous* operation, but intermittent operation is different. If the system samples a sensor now and calls a function to process the data immediately in the code, the sensor data might be hours old when it is processed, because power could fail before processing completes? A power failure every 100ms can leave the state of the system murky: "What was I doing before power failed?" Reasoning about program behavior is harder when power may fail any time.

Memory consistency is particularly challenging and is the focus of the NVMR paper. Parts of memory are non-volatile, and some dependencies can leave non-volatile memory (NVM) inconsistent when power fails. As the paper describes, if a program "persists" (i.e., commits to NVM) a value, but then reverts to an earlier checkpoint, NVM is left inconsistent. A read to that persisted location in a re-execution of the region sees the wrong value; similar problems arise if cache writebacks or checkpoint backups persist out of order. These subtle persistent ordering conditions all lead to incorrect behavior when a program re-executes from a checkpoint. Instead, regions between checkpoints should be *idempotent*, meaning re-execution always produces the same result.

Idempotence is where NVMR's solution to the persistent ordering problem comes in. The paper nicely characterizes these dependencies and shows that some lead to incorrect execution if left unaddressed. NVMR has a nice insight: *Rename* the memory locations being written, uniquely in each re-execution. That way, if power fails and a read in a re-execution attempts to access a location that was persisted in a prior execution attempt, the read does not see the persisted value and the program re-executes idempotently. NVMR is a set of hardware extensions that track memory accesses and persist actions. When a memory access could violate idempotence, NVMR's mapping table redirects the access to a different, safe NVM location. When remapped locations are no longer needed, NVMR reclaims their mapping state.

The simple technique in this paper works extremely well to avoid idempotence violations with low cost, because problematic accesses are replaced by idempotent writes to "fresh," renamed locations. The hardware cost is acceptable and the net effect of NVMR is to save 10% to 30% energy on average that would have gone to bookkeeping and moving data into and out of NVM. Energy is everything when you're running from a capacitor, so these savings translate to more capable intermittent devices.

The NVMR paper is part of the community's steady progression toward more capable intermittent computing devices with reasonable, basic correctness properties. These properties are now becoming formalized,[1] and the community is turning a corner to face the next great challenge in batteryless devices: energy efficiency. A concerning and even puzzling aspect of computing "at the low end," especially in intermittent systems, is that devices continue to be built around relatively *inefficient* von Neumann processors. Despite the addition of vector units, NPUs, DSP cores (and cool spoilers and racing stripes), these architectures are still hampered fundamentally by the inefficiency of instruction supply, control, and data movement in the von Neumann model. For the last decade of intermittent computing, including the great work in the NVMR paper, the focus was "How do we make an intermittent system do *anything useful*?" The success of systems approaches now clears the path for the future to be defined by a different question: "How much *more* can an intermittent system do?" This future will be defined by a new wave of energy-efficient architectures—RipTide (Gobieski), Monza (Efficient Computer), Ultra-Elastic CGRAS—that tap into what we have learned about correctness and bring to bear the benefits of efficiency to support intermittent applications that, even today, seem impossible. It is an exciting moment for the intermittent computing research community. C

References

1. Surbatovich, M. Designing formally correct intermittent systems. *Carnegie Mellon University* (2023);10.1184/R1/24123480.v1

Brandon Lucia (blucia@ece.cmu.edu) is the Kavčić-Moura Professor of Electrical and Computer Engineering at Carnegie Mellon University, Pittsburgh, PA, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

DOI:10.1145/3840403

To view the accompanying Technical Perspective, visit doi.acm.org/10.1145/3843283

Non-Volatile Memory Renaming for Safe Batteryless Computing

By Abhishek Bhattacharyya, Abhijith Somashekhar, and Joshua San Miguel

ABSTRACT

Batteryless devices powered by energy harvesting face frequent interruptions due to unstable power, requiring them to save their progress and resume later. These devices rely on non-volatile memories, such as Flash or FRAM, to preserve their state across power losses. However, saving data to these memories not only consumes significant energy, making frequent backups expensive, but can also lead to data inconsistencies and incorrect program execution when not handled appropriately.

This work introduces NvMR, an architecture that addresses these challenges by renaming non-volatile memory addresses to prevent inconsistencies. NvMR minimizes the need for backups, ensuring they occur only when absolutely necessary. It also simplifies the backup process, freeing it from program-specific constraints, and reduces energy consumption by 20%–40% compared to current state-of-the-art. These improvements make NvMR a significant step forward for energy-efficient and reliable operation in batteryless devices.

1. INTRODUCTION

By 2030, the proliferation of Internet of Things (IoT) devices is expected to reach staggering levels, with forecasts estimating more than 29 billion IoT-connected devices globally, growing at a compound annual rate of 13% from 2024.[21] This growth is driven by advances in connectivity technologies, such as 5G, low-energy Bluetooth, and sophisticated applications like machine learning, alongside expanding use cases in industrial automation, healthcare, smart homes, and urban infrastructure. A majority fraction of these devices relies on batteries, which are less eco-friendly because they wear out quickly due to frequent recharging. They are also inconvenient to replace in remote deployments, driving interest in *batteryless* devices that harvest energy from ambient sources, such as solar, RF, wind, or piezoelectric systems.[6,8,15,18]

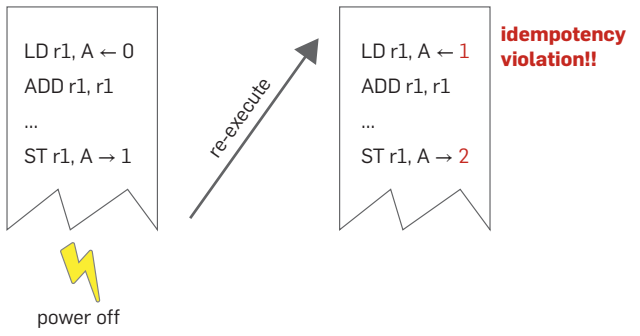
Batteryless devices store energy in supercapacitors (higher recharge cycles but less-dense energy storage) but face challenges due to unpredictable power supply. To ensure forward progress, they rely on *intermittent computing (IC)*, where the system backs up application and processor states to non-volatile memory (NVM) before energy depletes completely. Once recharged, the system restores state and resumes computation. The constraints of an IC system pose unique challenges that are not normally encountered in a continuously powered system.

The original version of this paper was published in *Proceedings of the Intern. Symp. on Computer* (June 2022).

1.1. Challenges in IC.

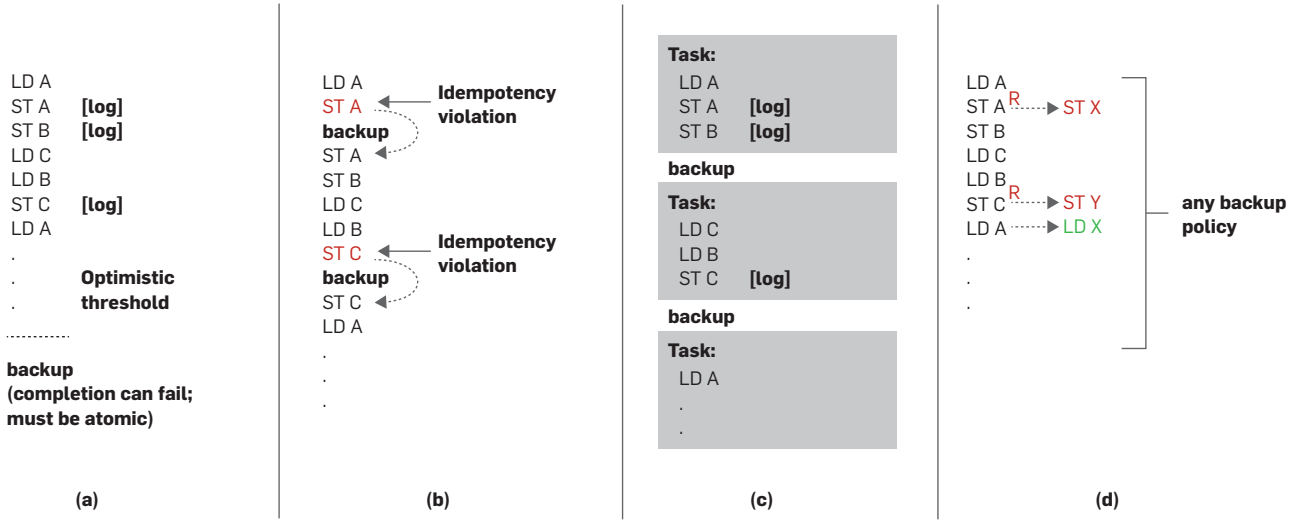
1.1.1. Inconsistency due to idempotency violations.
Unexpected power losses during program execution in batteryless devices can cause an incorrect processor state. To illustrate, in Figure 1 we show the execution of a loop with the iterator stored at NVM location `A`, the instruction `LD A` initially reads a value of 0 from address `A` and stores it in register `r1`. `ADDI` then increments the value in `r1` by 1, and `ST A` writes the value of `r1` back to address `A` in NVM. However, if a power outage occurs after the store, re-executing the program causes `LD A` to read the updated value of 1 from NVM and the subsequent `ST A` storing a value of 2, resulting in skipping an iteration, which can ultimately lead to incorrect behavior. This inconsistency in the result of `LD A` from the result prior to power failure is referred to as an *idempotency violation*, an artifact of write-after-read (WAR) dependency between `LD A` and `ST A`.

Figure 1. A sudden power loss can cause an idempotency violation in intermittent program execution.



1.1.2. Expensive conservative backups.
IC systems require backups to ensure data consistency during power failures. Backup schemes can be broadly classified as hardware- or software-based. Hardware-based approaches[2,10] use an analog-to-digital converter (ADC) to constantly monitor supply voltage and trigger a backup just before power failure. To ensure correctness, backups must be atomic (Figure 2a). Software-based methods[5,12] decompose programs into atomic tasks, with backups at task boundaries (Figure 2c). While these ap-

Figure 2. (a) Backup policy with an optimistic threshold (backup must be atomic), (b) Clank (backup when there is an idempotency violation), (c) DINO (backup at the boundary of predefined atomic tasks), (d) NvMR (any backup policy of choice); R stands for renaming. NVM addresses A and B are renamed to X and Y.



proaches ensure consistency, they often impose unnecessary backups due to conservative policies or static task decomposition.

Clank[9] and DINO[12] highlight the trade-offs in idempotency handling. Clank uses buffers to classify addresses as read-first or write-first, triggering backups on idempotency violations caused by store operations (Figure 2b). Although effective, this approach results in frequent backups, even when sufficient energy is available. Our experiments with Clank show that such violations account for more than 30% of total energy consumption in common embedded benchmarks. DINO, on the other hand, enforces task-level idempotency with privatization buffers, but static task decomposition limits flexibility and increases programmer effort. The challenge is to balance backups for correctness with forward progress. An ideal solution would ensure program correctness while optimizing energy efficiency.

We propose *NvMR*, an IC architecture that overcomes these challenges by dynamically renaming NVM addresses of violating persist stores (Figure 2d), decoupling backup policies from program characteristics, and allowing energy-efficient backup policies under varied conditions.

2. INTERMITTENT PERSIST DEPENDENCIES

Most of the previous works have built models to characterize the efficiency of IC, but none provide generalized models addressing the persistency constraints required for correct execution under intermittent power. Persist dependencies, essential for ensuring programs behave as if running on a continuously powered system, remain unexplored. While memory-persistency models[17] in shared-memory multiprocessors address orderings between persists to an external persistent data structure, this work introduces the first persist dependency model tailored for addressing the persist orderings imposed by IC.

Previous studies proposing correctness models for IC assume architectures with no dirty data buffering, such as write-through caches.[23] In contrast, we consider an architecture with a volatile write-back cache, which reduces costly NVM writes by reusing dirty data. Thus, there is a good chance that the persists for backups and stores can get reordered. Execution resumes from the last persisted backup after power loss. Any instructions executed after the backup must be re-executed. This work aims to provide a theoretical framework for understanding persist dependencies and their impact on intermittent execution.

2.1. Requirements for correct IC.

Correct execution in traditional intermittent architectures relies on three key requirements:

1. *Code Progress*. Stores to the same address must be persisted in program order, and backups must be persisted in the order they are triggered to ensure the program completes if the device sustains energy for at least one backup.
2. *Data Progress*. A store to NVM address X must be persisted before the backup following it, ensuring data remains up to date across power losses.
3. *Idempotency*. After a backup, if the first access to an address X is a load, subsequent stores to X must not be persisted until the next backup. This prevents data corruption from unbacked instructions after a power loss.

2.2. Persist ordering model.

To ensure correct execution in intermittent systems, it is essential to define the order of persist operations, including backups and stores. Current systems often enforce overly conservative constraints, limiting efficiency. We aim to characterize the *minimum persist dependencies* required for correctness, balancing flexibility and reliability. Specifically, *given a program and a set of backups invoked during its execution, what are the minimum ordering constraints for correct persists?* Each backup captures the

volatile processor state, including registers and the program counter, and persists it into NVM. Similar to stores, persisting the backup can be decoupled from its execution as long as the persisted register contents match what they would be at the position in program order at which the backup was triggered. The processor may trigger a backup and continue executing instructions after it; however, if power is lost before the backup is persisted, the processor must revert its state to an earlier backup. A backup must be persisted atomically via double buffering.

We formally describe persist dependencies using *happensbefore* relations between persist operations, expressed as {*p1*,*rel*,*p2*}:

- *p1* and *p2* are persist operations in program order (e.g., store or checkpoint).
- *rel* represents the persist order between a pair of persists with the order direction.

For example, $\{st, \xrightarrow{spo}, st\}$ is the relation between a pair of persist stores to the same address, representing the necessary ordering between them to maintain program order (Requirement 1). We color-code the various relationships that exist in an intermittent execution and list them alongside the requirements they satisfy in Table 1. The ordering, $\{st, \xrightarrow{spo}, st\}$, is also needed for correct program execution in continuously powered systems, while the other orderings are specific to IC systems. $\{backup, \xrightarrow{bpo}, backup\}$ ordering is necessary in multi-backup schemes only.[20]

Figure 3 shows the necessary persist dependencies for any general program execution, characterized by read-dominance and write-dominance, focusing on the symbolic NVM location X. We ignore the dependencies in instruction execution and assume they respect program order. We also ignore loads and assume the most up-to-date values are readily available in the write-back cache, except immediately after a power loss; in that case, a load must fetch the value from NVM. Backups can be invoked at any arbitrary point in the program, but when invoked, they introduce a set of happensbefore orderings and atomicity constraints with surrounding stores.

2.2.1. Understanding persist orderings in IC.

The backup point also decides if an address location X is read-dominated or write-dominated in the next intermittent code section.

Read-dominance. A location X is read-dominated if the first load after the most recent checkpoint occurs before the first store in program order. Figure 3(a) models the dependency pattern when all stores access read-dominated addresses. To ensure data progress (Requirement 2), each store must persist before the next backup persists. Otherwise, if the next backup were persisted first and power is lost, the most up-to-date value of X would be lost. Simultaneously, to ensure idempotent execution (Requirement 3), each store must not persist until the next backup persists. Otherwise, if the store were persisted first and power is lost, the program would re-execute from the prior backup and incorrectly load the later store's value of X. These constraints combined imply that all stores to a read-dominated address must atomically persist with the next backup. This is a unique, fundamental challenge in IC systems.

Write-dominance. A location X is write-dominated if the first store occurs before the first load. Figure 3(b) shows that for the same program, with a simple shift of when backups are invoked, X becomes write-dominated. As a result, the dependency pattern changes dramatically. Now only Requirements 1 and 2 need to be enforced, which is done by persisting backups in program order and ensur-

Table 1. Persist happens-before orderings (*spo*: store persist order, *bpo*: backup persist order, *rfpo*: read from persist order, and *irpo*: idempotent re-execution enforced persist order) in an intermittent execution, along with the correctness requirements they fulfill, shown with color-coded arrows.

Requirement	Persist Ordering
Code Progress	ST X → ST X (spo)
Code Progress	backup → backup (bpo)
Data Progress	ST X → backup (rfpo)
Idempotency	ST X → backup (irpo)

Figure 3. Happens-before persist orderings for correct intermittent execution. Cyclic sequence of orderings represents atomicity constraints.

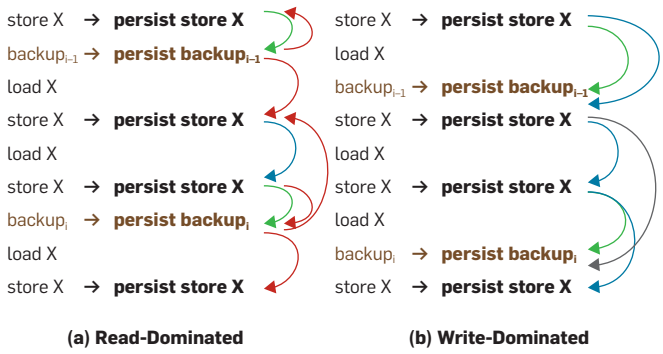
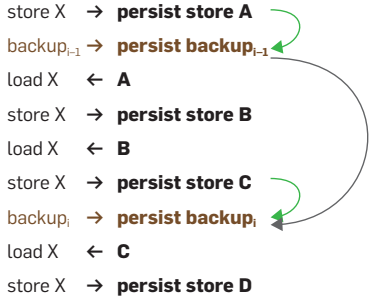


Figure 4. Happens-before persist orderings with proposed NVM renaming. By renaming X, all stores access only write-dominated memory locations. Only the stores that immediately precede backups must be persisted.



ing stores are persisted before the next backup. Stores no longer must wait until the next backup to persist, eliminating the atomicity constraint. In fact, stores after a backup may even be persisted before the backup persists. Since X is always write-dominated, whenever power is lost, regardless to which backup system it reverts, X will always be overwritten first, nullifying the last persisted store.

2.2.2. Why non-volatile memory renaming?

Idempotency is essential for correct intermittent execution and can only be violated by WAR dependencies on NVM locations. Our architecture addresses this by dynamically *renaming* read-dominated NVM addresses, inspired by similar techniques used in out-of-order processors for registers. *NVM renaming* ensures idempotent execution (Requirement 3) regardless of backup timing.

The key insight is: *renaming transforms all NVM addresses into write-dominated ones.* As shown in Figure 4, renaming eliminates dependencies such as $\{st, \xrightarrow{spo}, st\}$, $\{backup, \xrightarrow{bpo}, st\}$, and $\{st, \xleftarrow{irpo}, backup\}$ from Figure 3(a).

This approach achieves the theoretical maximum efficiency by only persisting stores immediately preceding a backup, satisfying Requirement 2 (data progress). Backups must still be persisted in the order invoked to ensure Requirement 1 (code progress). Thus, renaming minimizes persist constraints, allowing backup decisions to be decoupled from program correctness requirements.

Figure 5. Block diagram of NvMR architecture (blocks shown in gold are added).

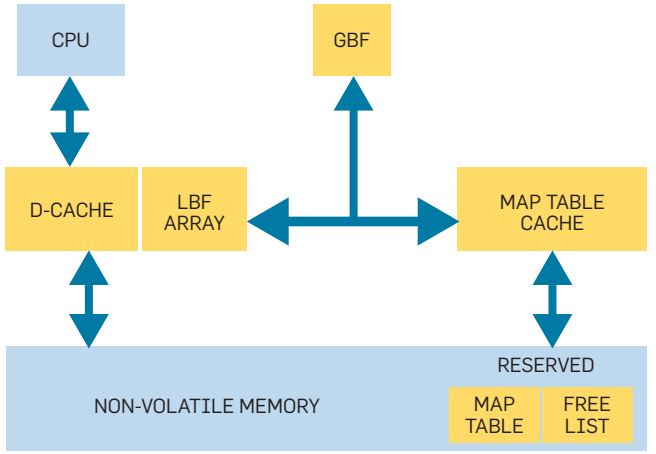
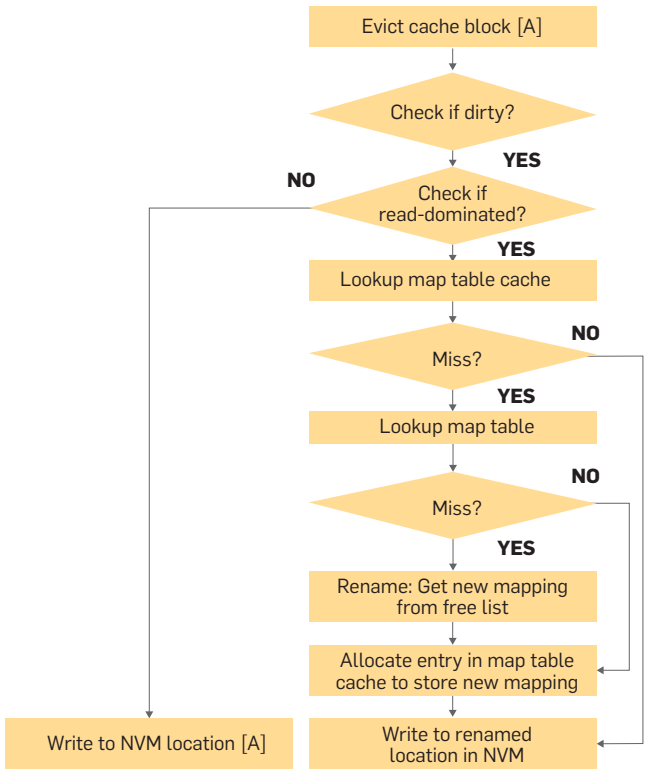


Figure 6. Flowchart showing the handling of idempotency violations in NvMR.



3. NvMR ARCHITECTURE

Based on insights from our previous analysis, we propose an intermittent architecture, *non-volatile memory renaming* (*NvMR*), that efficiently detects idempotency violations due to persist stores in hardware and renames their NVM addresses. Figure 5 is a high-level block diagram of NvMR. *Map table* and *free list* are two software data structures allocated statically in NVM at compile time. Map table keeps a record of the old (original or most recently persisted) mappings. Free list is a queue of available mappings. Four hardware components are important for NvMR's renaming mechanism:

1. Write-back, write-allocate (WBWA) data cache that exploits data locality and tracks dirty blocks
2. Global bloom filter (GBF) that logs the composite state (Unknown=0 or Read-dominated=1; discussed later) of a block when it is evicted from the cache
3. Local bloom filter (LBF) array, tightly coupled with cache data array, with one LBF per block that logs the state of each word in a block (Unknown=00, Read-dominated=01, and Write-dominated=10)
4. Map table cache, a volatile cache for map table (analogous to a translation lookaside buffer for page table) that saves energy and time by reducing the number of trips to NVM. Apart from the old mapping, it also holds the new (latest) mapping. Since it is also a WBWA cache, for each block there is a dirty bit along with a valid bit and a tag.

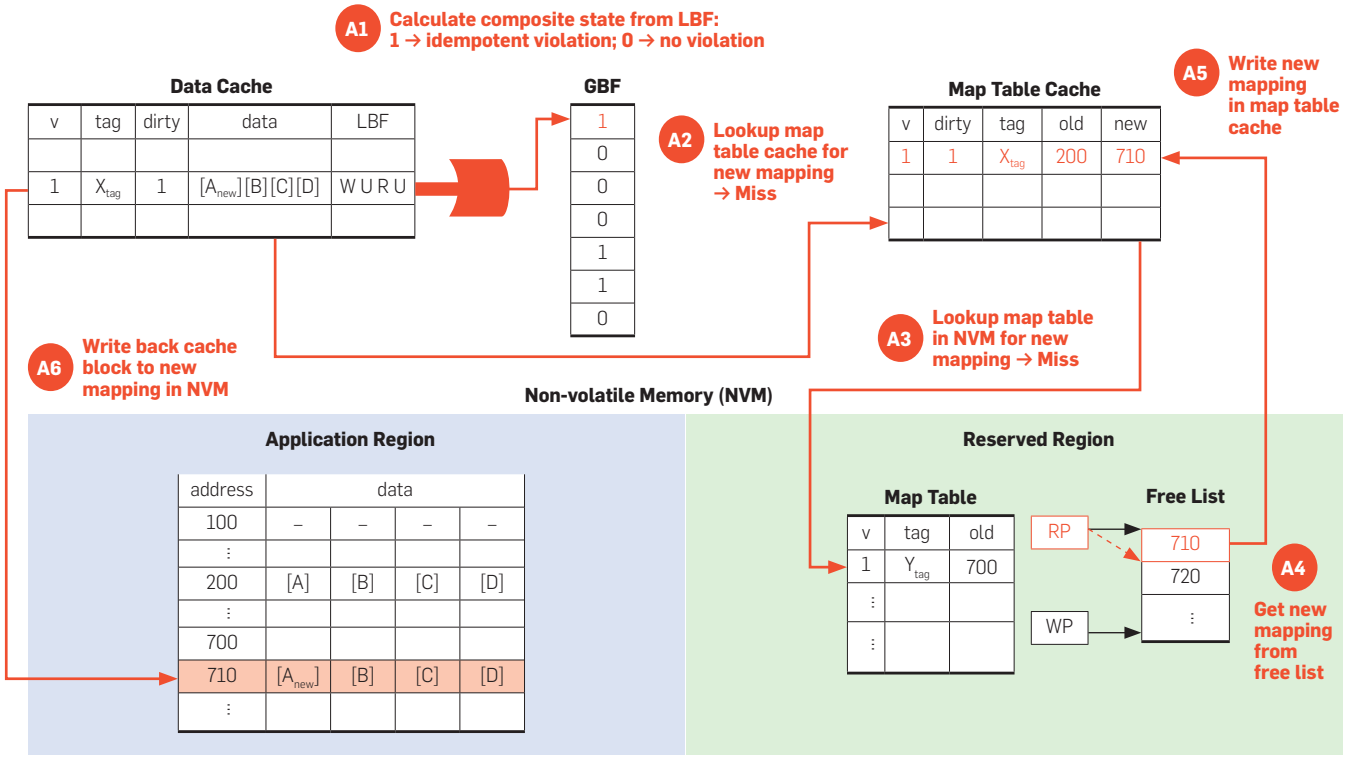
3.1. Intermittent execution in NvMR.

Figure 6 explains NvMR's renaming mechanism in a concise flowchart. To explain NvMR's microarchitecture in more detail, we divide intermittent execution in NvMR into five disparate events and explain them with a running example.

3.1.1. Idempotency violation.

Figure 7 shows a cache block X with a tag, X_{tag}. This block was fetched from NVM because of a read-dominated access to word [C]. Later, as word [A] is updated to [A_{new}], the block is marked dirty and the corresponding LBF states of [C] and [A_{new}] are set as R and W respectively. Only when

Figure 7. Eviction of a cache block with tag X_{tag} causes an idempotency violation. Cache block address 200 is renamed to address 710. The dirty cache block is stored at the new mapping 710 instead of 200.



a dirty block is evicted is the composite state of the block calculated by OR of the LSBs of the LBF states of the constituent words. If the composite state is 0 (block is write-dominated), there is no idempotency violation. Otherwise, it should be flagged as an idempotency violation since it is dirty and read-dominated (violates Requirement 3). Thus, NvMR flags an idempotency violation (A1) when block X is evicted. The composite state is logged in GBF.

3.1.2. Renaming.
Eviction of an idempotency violating block triggers a lookup in the map table cache to find its new mapping. In case of a hit, the new mapping is used to persist the block. In case of a miss, map table is looked up. If the map table lookup is successful, the old mapping is fetched into the map table cache. Otherwise, a new mapping is popped from the head of the free list and fetched into the map table cache. The block is persisted using the fetched mapping. Thus, the old mapping of the persist store is renamed to a new mapping. In Figure 7, block X incurs a miss in both map table cache (A2) and map table (A3). Thus, NvMR fetches mapping 710 from free list (A4) and inserts a new dirty map table cache block (A5), renaming 200 to 710. The contents of block X are written to the renamed address 710 (A6).

3.1.3. Cache miss.
In Figure 8, a store request for cache block X attempts to write [B_{new}]. Since the block was recently evicted, the lookup results in a cache miss (B1). The cache controller then looks up the map table cache to find the latest mapping in NVM. As address 200 was renamed to 710 and this entry exists in the map table cache, the lookup is a hit (B2). The block is fetched from 710 in NVM, updated with [B_{new}], and marked dirty (B3). NvMR then checks GBF to determine the block state. Since block X is read-dominated (1), all associated LBF states are set to R (B4). For blocks in the U state, LBF states are marked R or W based on the type of request. A map table cache miss triggers a lookup in the map table, which lies in the critical path. We assume the map table lookup can be done in parallel to fetching the application data, since both reside in separate regions of NVM.

3.1.4. Backup.
NvMR initiates a backup in three cases: (1) when a dirty map table cache entry is evicted to ensure the map table in NVM reflects the latest mappings, (2) based on a user-defined backup policy to guarantee forward progress, and (3) when an idempotency violation occurs and either the map table is full or the free list is empty, making renaming impossible. Like other intermittent architectures, NvMR snapshots general-purpose and special registers (e.g., the program counter), but these are omitted in Figure 9. With a write-back data cache, NvMR persists all dirty blocks during backup. If a cache block was renamed, NvMR fetches its latest mapping from the map table cache or the map table; otherwise, the block is stored at its program address.

For cache block X, the map table cache lookup is a hit

Figure 8. A store miss causes the cache block to be fetched from the new mapping 710. NvMR keeps a log of the new mapping in the map table cache.

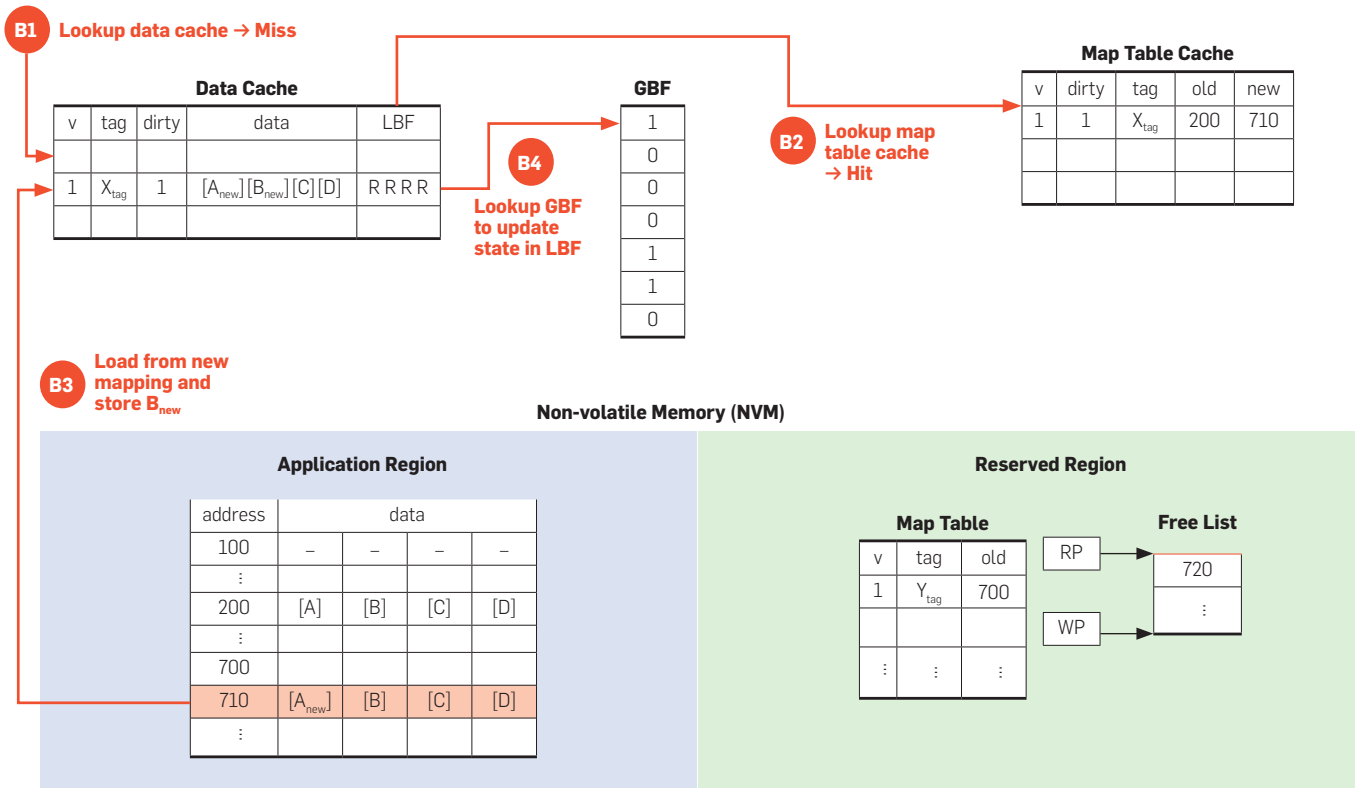
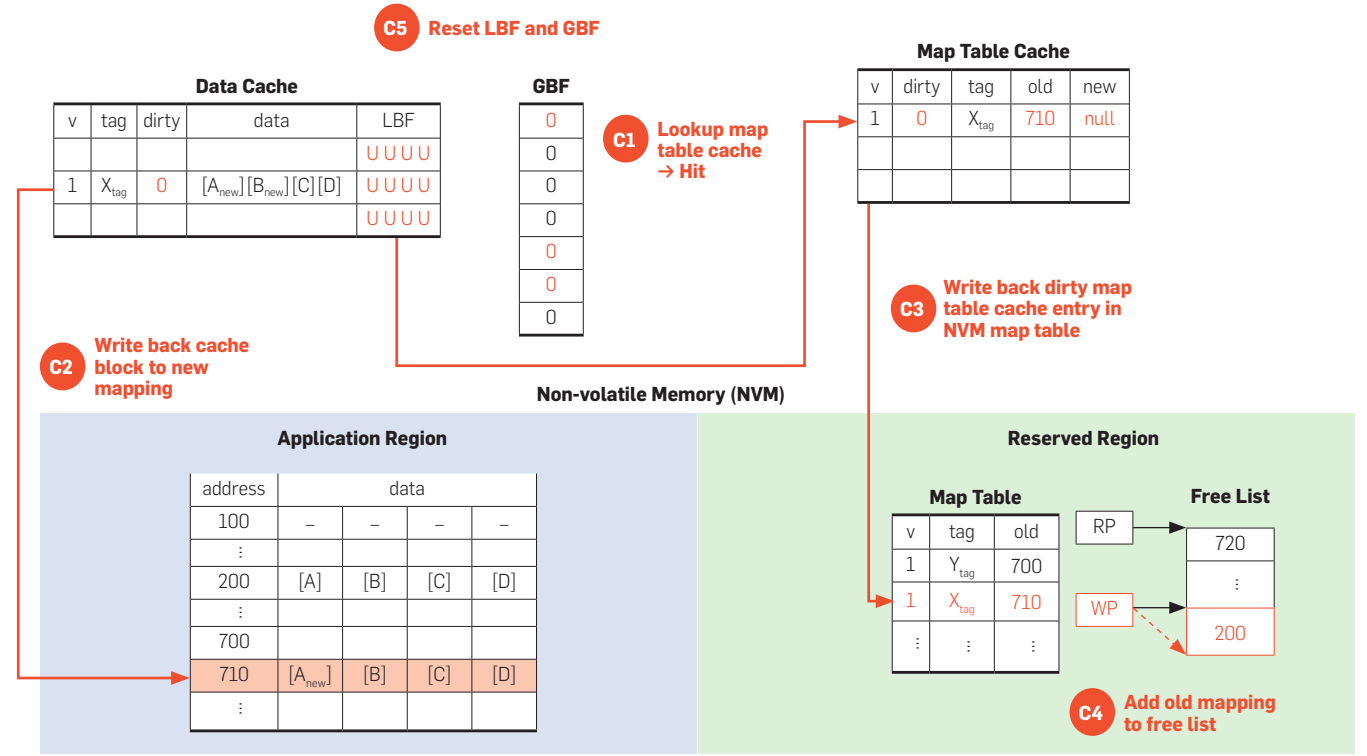


Figure 9. Backup will cause the data cache to persist its dirty blocks. The old mappings from the map table cache are pushed into the free list, and the new mappings are written to the map table. All states in LBF and GBF are reset.



(C1), and the dirty data is persisted at the new mapping 710 (C2). The dirty bit of block X is cleared after the write-back. NvMR writes the new mapping 710 to the map table (C3) and appends the old mapping 200 to the free list tail (C4), indicated by the orange arrow for the write pointer. Once all updates are completed, the new mapping is copied to the old field in the map table cache, and the dirty bit is reset. Finally, GBF and LBF arrays are cleared (C5).

3.1.5. Restore.

After a power loss, an IC system state is restored to the most recent backup and execution resumes. In NvMR, restore is pretty simple: The program counter and other registers are restored from the most recent snapshot saved in NVM. Moreover, only the read and write pointers of the free list need to be restored.

3.1.6. Reclamation.

The map table, being a limited hardware resource, can fill quickly for applications with frequent idempotency violations, especially in energy-harvesting devices with small non-volatile memory. To address this, we introduce *reclamation* (a popular technique in garbage collection), which reuses the original mapping—the program's cache block address. When the map table is full after a backup, NvMR reclaims an entry using a cache-like replacement policy. This involves atomically moving the old mapping to the free list's tail, invalidating the entry, and copying data from the freed mapping back to the original mapping. NvMR also invalidates the corresponding map table cache entry on a lookup hit. Reclaiming reduces backups after a full map table but decreases the free list's effective size, as original mappings cannot be added. To compensate, the free list requires more mappings to avoid shortages.

4. METHODOLOGY

We extend a cycle-accurate C++ simulator[20] to model an intermittent ARM Cortex M0+[1] (3-stage in-order pipeline) executing ARM Thumb ISA instructions. Voltage traces sampled at 1kHz from prior work[14] are used to compute the charge across the supply supercapacitor. Dynamic and leakage power for added volatile components are modeled using CACTI,[16] and NVM energy values are taken from the STM32L011K4 MCU data sheet.[22]

We compare NvMR with two IC architectures: a) a modified *Clank*[9], augmented with local and global bloom filters, and a write-back data cache, an idempotency violation-aware hardware solution, and b) *HOOP*[4], a software transactional memory solution with hardware support with similar semantics to a task-based IC architecture. We use the configuration listed in Table 2 for all our results except for the sensitivity analysis because it is pareto-optimal. Since NvMR decouples the backup policy from the memory access patterns of an application, we show the impact of NvMR using three backup schemes:

- *Just-In-Time* (JIT) is an oracle that accurately estimates when a power loss will happen and triggers a backup just before it.
- *Spendthrift* uses a lightweight neural network (trained offline with two inputs, available ambient energy, and supercapacitor voltages at a particular time to an accuracy of 97%) to predict backups,[13] representative of JIT schemes deployed commercially.
- *Watchdog timer* is the most naive out of the three and invokes a backup every *8,000* cycles.[9]

Table 2. System configuration used in evaluation.

Parameter	Value
Processor	ARM Cortex M0+ (Thumb ISA), 8MHz
Data Cache	256B, 8-way associative, 16B block, LRU, 1-cycle hit latency
GBF	8 one-bit entries
LBF	4 two-bit entries per cache line
Map Table Cache	512 entries, 8-way associative, LRU
Map Table	4,096 entries, LRU
Free List	4,096 + 512 + 1 = 4,609 mappings
Flash	2MB
Supercapacitor	100mF, 2.4V max. voltage

The benchmarks we chose are from MiBench[7] and PERFECT[3] benchmark suites, representing applications typically found in ultra-low-power IoT systems with a high number of idempotency violations.

5. EVALUATING NvMR

To understand the benefits offered by NvMR, we compare it with the chosen baseline IC architectures by looking at the energy savings in each architecture with different backup schemes. In addition, we analyze how sensitive NvMR is to variations in system configuration. Further, we discuss the minimal overheads incurred with NvMR. All results are averaged over 10 voltage traces.

5.1. NvMR vs. Clank.

Figure 10 presents the percentage of energy saved in NvMR compared to Clank. With JIT, NvMR saves on average about 20% energy compared to Clank. The energy savings can range from 2% to 37% based on the application. With spendthrift, NvMR saves about 15.6% energy on average. Since *Blowfish* and *dijkstra* spend the least amount of energy on backups due to idempotency violations, they marginally consume more energy in NvMR. With watchdog timer, NvMR saves on average 9% energy. *stringsearch* and *hist* perform worse than in Clank because a significant portion of energy (6.5%–7%) is wasted in re-executing instructions after a power failure (dead energy[20]).

We take a deeper dive by analyzing the breakdown of normalized energy consumption of Clank and NvMR with JIT backup scheme in Figure 11. The total energy consumption is split into four components—*forward progress, backup, restore,* and *dead* energy.[20] NvMR incurs additional energy overhead for forward progress and restore due to dynamic and leakage energy from the map table cache, extra NVM accesses for the map table and free list, and reclamation-related energy.

JIT backup scheme eliminates dead energy, and NvMR's restore energy is minimal, involving only the retrieval of

Figure 10. Energy saved in NvMR with different backup schemes compared to Clank.

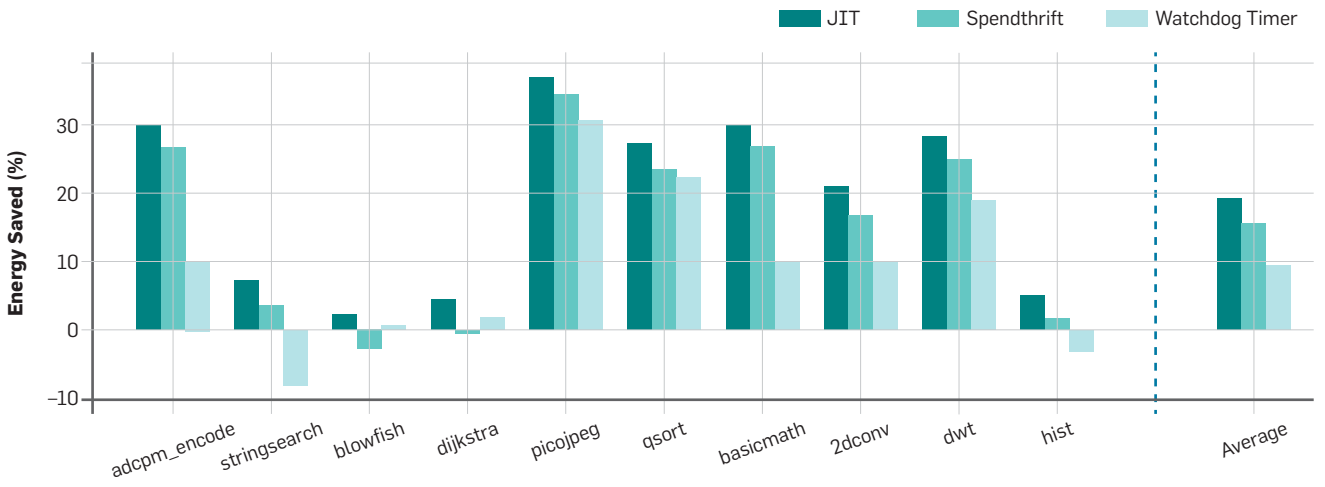


Figure 11. Breakdown of normalized energy consumption of Clank (left bars) and NvMR (right bars).

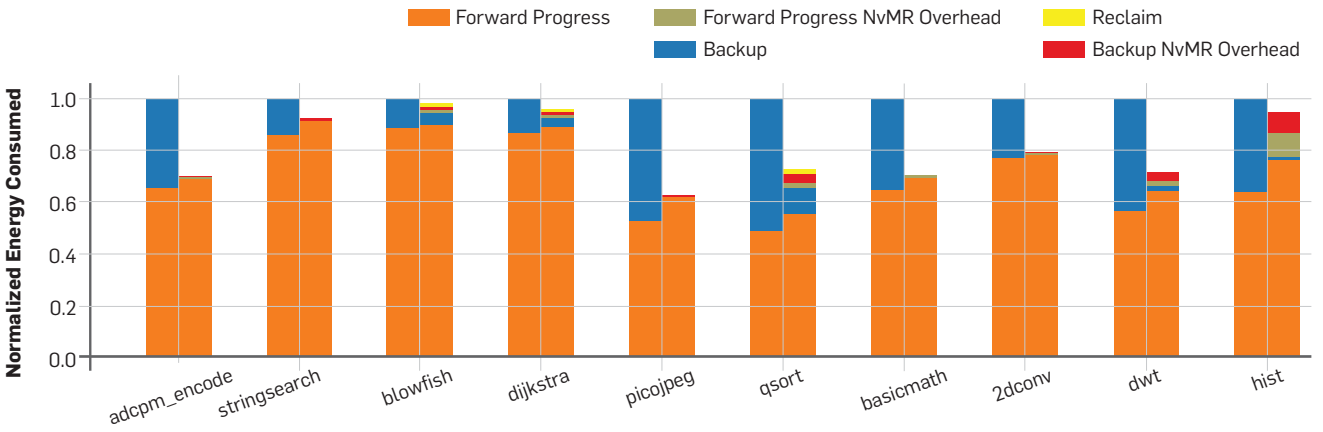
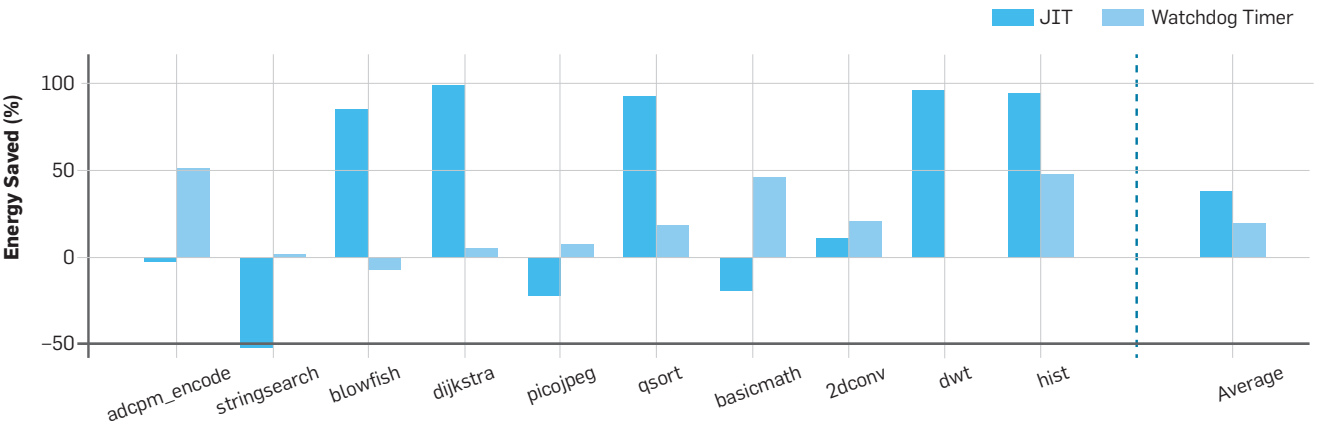


Figure 12. Energy saved in NvMR compared to HOOP.



CPU registers; hence, these are omitted from Figure 11. The benchmark *stringsearch* performs worst, with nearly 90% of its energy spent on forward progress due to minimal idempotency violations during string list searches. NvMR incurs forward progress and backup overheads, along with additional energy for reclaiming. Forward progress energy is higher in NvMR than Clank for some benchmarks, since persisting data to renamed addresses is included in forward progress rather than backup energy. Reclaim energy is significant only for benchmarks such as *blowfish*, *dijkstra*, and *qsort*, where frequent idempotency violations fill the map table. Overall, renaming and reclaiming add just 3% to NvMR's total energy, a small cost for reducing backups by 185x on average.

5.2. NvMR vs. Hoop.

HOOP, a log-based architecture, performs out-of-place updates to an NVM region, optimizing write efficiency and triggering backups when its buffer or mapping table fills up. Table 3 outlines the HOOP configuration used for the results in Figure 12. Components, excluding the mapping table (idealized as infinite for no energy usage), are sized to match NvMR's on-chip area and memory overheads (Table 2). With JIT, NvMR saves 40% more energy on average, except for benchmarks *stringsearch, picojpeg*, and *basicmath*, where HOOP excels due to high store locality, allowing its buffer to reduce NVM writes. For watchdog timer backup, NvMR achieves 19.4% average energy savings, although less than with JIT due to the naive nature of the timer scheme. (Spendthrift results are not shown since they were not part of the original paper.)

Table 3. System configuration of a simplified version of HOOP. The infinitely large map table has no energy and area overhead.

Structure	Mapping Table	OOP Buffer	OOP Region
Size	Infinite	128	2048

5.3. NvMR sensitivity.

Map Table Cache. Energy savings in NvMR improve with larger map table caches, as seen in Figure 13b, since evictions of dirty entries (and the associated backups) reduce. Associativity has minimal impact on energy efficiency after reaching four-way; savings plateau at around 0.2% with full associativity, as shown in Figure 13a. These results use a map table with 4,096 entries and JIT backup for consistency.

Map Table Size. Increasing the map table entries from

Figure 13. Energy saved with different map table cache and map table configurations and capacitor sizes.

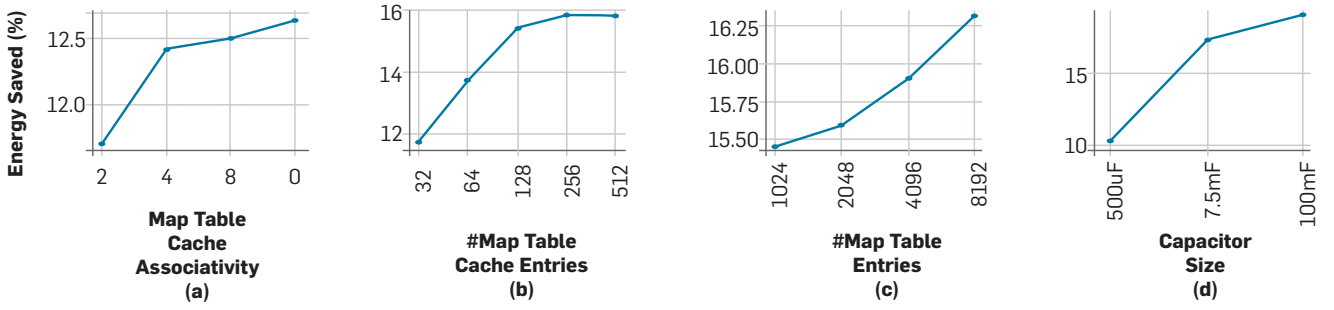
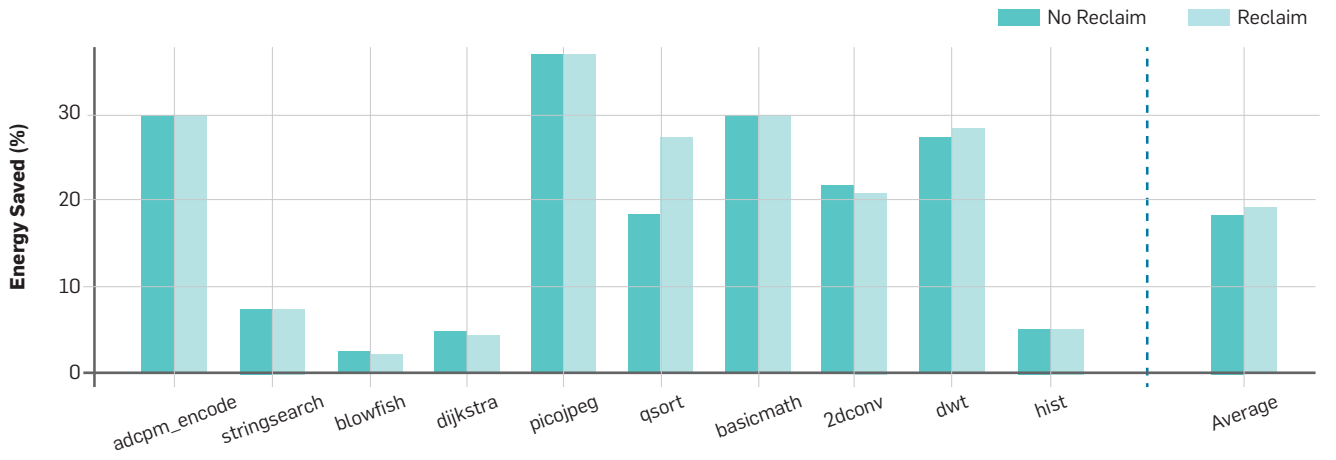


Figure 14. Energy saved in NvMR with and without reclaim.



1,024 to 8,192 shows a modest 1% energy saving (Figure 13c), primarily benefiting benchmarks prone to idempotency violations, such as *blowfish, dijkstra, qsort*, and *dwt*. Given the large Flash requirements for small gains, 4,096 entries represents a balanced choice, aligning with the constraints of energy-harvesting devices.

Capacitor Size. NvMR's energy savings grow as capacitor size increases, with a 7% improvement from 500µF to 7.5mF and a slower 2% rise up to 100mF (Figure 13d). Larger capacitors extend active periods, leading to more idempotency violations, which NvMR exploits to reduce backups. However, the rate of savings diminishes beyond 7.5mF, reflecting diminishing returns on larger capacitor sizes.

5.4. Benefits of reclamation.

Reclamation minimizes backups caused by idempotency violations when the map table is full, but only a few benchmarks encounter this issue with the configuration in Table 2. As shown in Figure 14, reclaiming provides minimal energy savings, averaging just 1%, without diminishing NvMR's overall efficiency *qsort* (9%) and *dwt* (1%) benefit, while *dijkstra*, *blowfish*, and *2dconv* incur slightly higher energy consumption. Nonetheless, reclamation can be particularly valuable for devices with limited NVM, where allocating large map tables and free lists is impractical. Our experiments reveal with a map table of 1,024 entries, reclaiming saves about 9% more energy compared to the "no reclaim" scenario.

5.5. NvMR overhead.

The hardware cost of NvMR is limited to the map table cache, with McPAT[11] estimates indicating a modest 6% on-chip area overhead for the configuration in Table 2. NvMR's software and hardware structures can be tailored to application requirements, constrained only by the available memory onboard. The memory footprint for the reserved region is approximately 6% of the 2MB Flash (typical size of Flash in commercial MCU boards[19]). However, if the NVM is byte-addressable, NvMR's renaming mechanism provides a significant advantage by reducing wear-out, achieving an impressive 80.8% average decrease in maximum writes per location compared to Clank.

Conclusion

This work addresses a critical challenge in energy-harvesting intermittent systems: backups caused by idempotency violations. While prior research has improved backup efficiency, unnecessary overhead from these violations persists. Our key insight is that backups should stem solely from energy or hardware constraints, not program inconsistencies. By analyzing unique persist dependencies in intermittent execution, we show that renaming NVM addresses of violating persists minimizes backups. Based on this, we propose NvMR, an energy-efficient hardware architecture that enables productive use of harvested energy and decouples program correctness from backup policies, paving the way for advancements in intermittent system design. C

References

1. ARM Ltd. *Cortex-M0+ Technical Reference Manual — Arm Developer*, 12 (2012).
2. Balsamo, D. et al. Hibernus: Sustaining computation during intermittent supply for energy-harvesting systems. *IEEE Embedded Systems Letters* (2015).
3. Barker, K. et al. *PERFECT (Power Efficiency Revolution for Embedded Computing Technologies) Benchmark Suite Manual*. Pacific Northwest National Laboratory and Georgia Tech Research Institute (Dec. 2013); http://hpc.pnnl.gov/projects/PERFECT/.
4. Cai, M., Coats, C.C., and Huang, J. Hoop: Efficient hardware-assisted out-of-place update for non-volatile memory. In *Proceedings of the 2020 ACM/IEEE 47th Annual Intern. Symp. on Computer Architecture* (2020), 584–596.
5. Colin, A. and Lucia, B. Chain: Tasks and channels for reliable intermittent programs. In *Proceedings of the 2016 ACM SIGPLAN Intern. Conf. on Object-Oriented Programming, Systems, Languages, and Applications* (2016), 514–530.
6. Gollakota, S., Reynolds, M.S., Smith, J.R., and Wetherall, D.J. The emergence of RF-powered computing. *Computer 47*, 1 (2014), 32–39.
7. Guthaus, M.R. et al. MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of the IEEE Intern. Workshop on Workload Characterization* (2001), 3–14.
8. Hester, J., Storer, K., and Sorber, J. Timely execution on intermittently powered batteryless sensors. In *Proceedings of the 15th ACM Conf. on Embedded Network Sensor Systems*, ACM (2017).
9. Hicks, M. Clank: Architectural support for intermittent computation. In *Proceedings of the ACM/IEEE Intern. Symp. on Computer Architecture* (2017), 228–240.
10. Jayakumar, H., Raha, A., Lee, W.S., and Raghunathan, V. Quick recall. *ACM J. on Emerging Technologies in Computing Systems 12*, 1 (2015), 1–19.
11. Li, S. et al. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM Intern. Symp. on Microarchitecture* (2009), 469–480.
12. Lucia, B. and Ransford, B. A simpler, safer programming and execution model for intermittent systems. In *Proceedings of the 36th ACM SIGPLAN Conf. on Programming Language Design and Implementation*, ACM (2015), 575–585.
13. Ma, K. et al. Spendthrift: Machine learning based resource and frequency scaling for ambient energy harvesting nonvolatile processors. In *Proceedings of the 2017 22nd Asia and South Pacific Design Automation Conf.* (Jan. 2017), 678–683.
14. Matthew Furlong, J. H. Persistlab/batterylesssim (Nov. 2017).
15. Merrett, G.V. and Al-Hashimi, B.M. Energy-driven computing: Rethinking the design of energy harvesting systems. In *Proceedings of the 2017 Design, Automation and Test in Europe*, 2017.
16. Muralimanohar, N., Balasubramonian, R., and Jouppi, N. Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0. In *Proceedings of the Annual Intern. Symp. on Microarchitecture* (2007), 3–14.
17. Pelley, S., Chen, P.M., and Wenisch, T.F. Memory persistency. In *Proceedings of the 41st Annual Intern. Symp. on Computer Architecture*, IEEE (2014), 265–276.
18. Ransford, B., Sorber, J., and Fu, K. Mementos: System support for long-running computation on RFID-scale devices. In *Proceedings of the 16th Intern. Conf. on Architectural Support for Programming Languages and Operating Systems*, ACM (2011), 159–170.
19. Renesas Microcomputers. *Arm® Cortex®-M0+ Ultra-low Power MCU Based on SOTB™ Process Technology*, *5* (2020). Rev. 1.10.
20. San Miguel, J. et al. The EH Model: Early design space exploration of intermittent processor architectures. In *Proceedings of the ACM/IEEE Intern. Symp. on Microarchitecture* (2018).
21. State of IoT 2024: Number of connected iot devices growing 13% to 18.8 billion globally. (2024); https://iot-analytics.com.
22. STMicroelectronics. *Access line ultra-low-power 32-bit MCU Arm®-based Cortex®-M0+, up to 16KB Flash, 2KB SRAM, 512B EEPROM, ADC*, (2017).
23. Surbatovich, M., Lucia, B., and Jia, L. Towards a formal foundation of intermittent computing. In *Proceedings of ACM Program. Lang. 4* (Nov. 2020).

Abhishek Bhattacharyya (abhattacha24@wisc.edu) is a Ph.D. student at the University of Wisconsin-Madison, Madison, WI, USA.

Abhijith Somashekhar, who was a student at the University of Wisconsin-Madison, Madison, WI, USA, at the time this research was conducted, is now a CPU performance engineer at Apple Inc., San Francisco, CA, USA.

Joshua San Miguel is an associate professor at the University of Wisconsin-Madison, Madison, WI, USA.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

University of Macau
Full/Associate/Assistant Professor in Human Computer Interface/ Data Science

The Institute of Artificial Intelligence and Brain Sciences (IAIBS) of the University of Macau invites applications for the positions of Full/ Associate/Assistant Professor in the following disciplines. These positions are intended to complement and strengthen existing faculty staff expertise. We are seeking candidates with a proven track record in scientific research, an innovative research vision, and a commitment to education.

1. Human Computer Interaction (Ref. No.: IAIBS/FAAP/HCI/04/2026)

Prospective applicants should specialize in human computer interaction, cognitive neuroscience, emotional AI, or the interface of brain sciences and AI in general and should possess a PhD and/or MD-PhD with extensive experiences in related fields. Preference will be given to candidates with advanced expertise and in-depth knowledge of neuroimaging and brain intervention.

2. Data Science (Ref. No.: IAIBS/FAAP/ DS/07/2026)

Prospective applicants should have a PhD degree in Artificial Intelligence, Computer Science, Data Science, or related disciplines. Outstanding candidates in closely related areas of interdisciplinary AI and data science are also encouraged to apply: (1) AI for Science; (2) AI for Society; and (3) AI Governance.

Applicants should visit https://career.admo.um.edu.mo/ for more details and apply ONLINE.



ADVERTISING IN CAREER OPPORTUNITIES

How to Submit a Classified Line Ad: **Send an e-mail to *acmmediasales@acm.org*. Please include text, and indicate the issue/or issues where the ad will appear, and a contact name and number.**

Estimates: **An insertion order will then be e-mailed back to you. The ad will by typeset according to CACM guidelines. NO PROOFS can be sent. Classified line ads are NOT commissionable.**

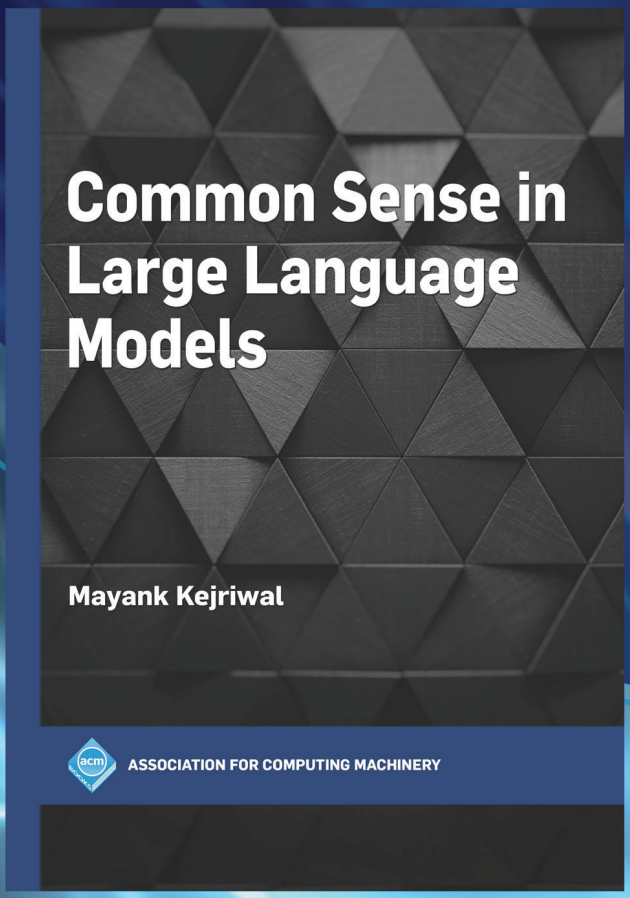
Deadlines: **20th of the month/2 months prior to issue date. For latest deadline info, please contact: *acmmediasales@acm.org***

Career Opportunities Online: **Classified and recruitment display ads receive a free duplicate listing on our website at: *http://jobs.acm.org***

Ads are listed for a period of 30 days.

For More Information Contact:
ACM Media Sales
at 212-626-0686 or
acmmediasales@acm.org

NEW BOOK RELEASE



ACM BOOKS
Collection IV

Common Sense in Large Language Models

by Mayank Kejriwal

ISBN: 979-8-4007-3198-3
DOI: 10.1145/3789270

While modern large language models (LLMs) like ChatGPT and Gemini appear deceptively capable in dialogue, they still occasionally fail at the intuitive physical and social tasks that define human common sense. This text synthesizes decades of research and offers an examination of the structural divides.

This book provides the first comprehensive scientific examination of the structural divide between massive knowledge acquisition and the fluid "common sense" reasoning required for human-level intelligence in machines. Written for researchers, artificial intelligence practitioners, and graduate students, this volume offers a roadmap for building capable intelligence by considering the strengths and limitations of large language models through the important lens of machine common sense–a grand challenge since the founding days of AI in the 1950s. It serves as an essential reference for advanced seminars seeking to bridge the conceptual gaps between fields like cognitive science, the history of computing, and modern-day AI. It also suggests ideas and frameworks for developing the next generation of reasoning-capable machines.

http://books.acm.org

[CONTINUED FROM P. 112] strategy Configuration 2.

For example, suppose X = 10 and X+d = 80. B places its candidates based on Configuration 2 at 11 and 79 and get roughly 34 each. Those two enter the runoff. B places another candidate at 81. A's candidate at 10 will get 10% of the votes and A's candidate at 80 will get 1% of the vote.

To show this works in general, note that party A has every interest to make *X* and *X+d* symmetrically distant from the ends, so *X = 100 - (X+d)*. Otherwise, in either configuration, party B will take the bigger end (automatically in configuration 1 and by choice in configuration 2). This symmetry implies that *2X = 100-d*. Recall that configuration 1 wins for party B provided *d/4 < X-1*. Given the symmetry condition, *d/4 < ((100-d)/2) - 1 = 49 - d/2*. So *d < (4/3) x 49 = 66*. For example, if *d = 64*, then party A places its candidates at *X = 18* and *X+d = 82*. Party B places its candidates, under the configuration 1 strategy, at 17, 83, and 50. The candidate at 50 obtains all votes from 35 to 65. The A candidate at 18 obtains 18 through 34 inclusive or 17%. The A candidate at 82 will obtain from 66 to 82, also 17%. However the B candidate at 83 will obtain 18% and the B candidate at 50 will obtain 31% (35 to 65). So the B candidates will appear in the runoff.

The elections work in two rounds. At the end of the first round, the two candidates who receive the most votes face each other in a runoff.

Configuration 2 wins for party B if *X < (d/2) - 1*. So, for example, if *d > 53*, then Configuration 2 wins. In the case that *d = 54*, party A places its candidates at *X = 23* and *X+d = 77*. Party B places candidates at 24, 76, and 78. The party B candidates at 24 and 76 each obtain 26\% of the vote. The party A candidate at 24 obtains only 24% of the votes and the party A candidate at 76 obtains only 1% of the vote.

Thus, if *d > 64*, use configuration 1. Otherwise use configuration 2. In fact both configurations will work for *d* between 54 and 64, inclusive.

End of Solution

Upstart: Suppose that party X has C candidates and party Y has C' candidates and C > C', but X must place all its candidates before Y. For which values of C and C' can Y still guarantee to field at least one candidate in the runoff? Are there any values for which Y can guarantee to field two candidates in the runoff? C

Dennis Shasha (shasha@cs.nyu.edu) is a professor of computer science in the Department of Computer Science at the Courant Institute at New York University, New York, NY, USA, as well as the chronicler of his good friend the omniheurist Dr. Ecco.

All are invited to submit their solutions to upstartpuzzles@cacm.acm.org; solutions to Upstarts and discussion will be posted at http://cs.nyu.edu/cs/faculty/shasha/papers/cacmpuzzles.html.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2026 Copyright held by the owner/author(s).

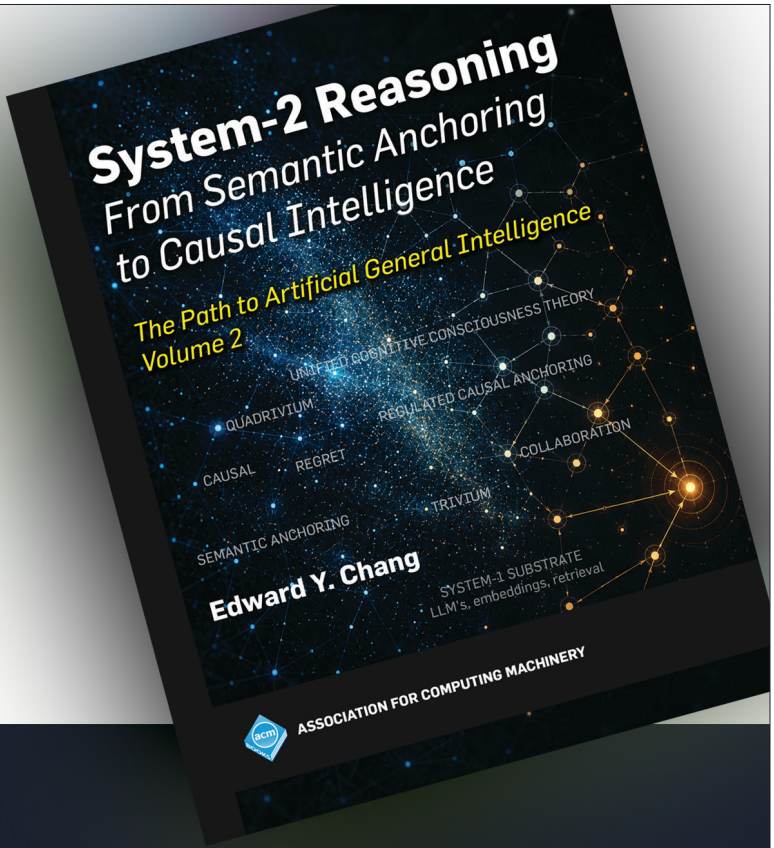
System-2 Reasoning

From Semantic Anchoring to Causal Intelligence: The Path to Artificial General Intelligence, Vol 2

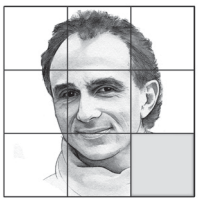
Edward Y. Chang

ISBN: 979-8-4007-2805-1
DOI: 10.1145/3822380

http://books.acm.org



ACM BOOKS
Collection III



DOI:10.1145/3837872

Dennis Shasha

Upstart Puzzles
Cheating by the Rules

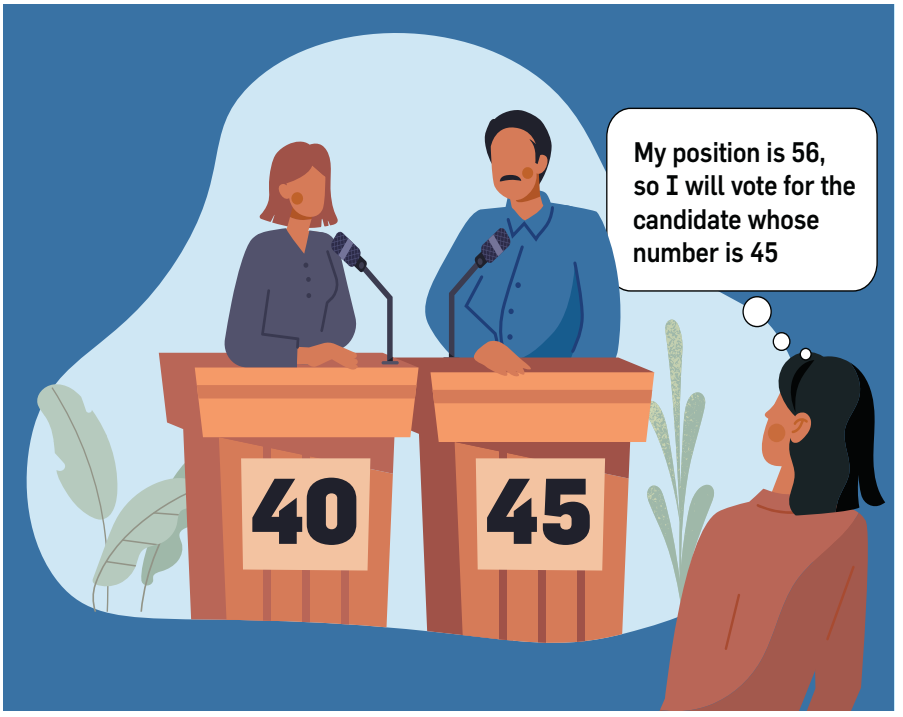
Calculating candidate options.

SUPPOSE THE POLITICAL spectrum is analogous to a measuring stick numbered between 1 and 100 (where 1 is totalitarian and 100 is anarchy, with liberal democratic being approximately 50), Here are the rules: Any candidate must state his or her integral position value (some whole number between 1 and 100, inclusive) and no two candidates may choose the same value. Voters will vote for the candidate nearest their position. If there are two candidates at the same distance from a voter V, then voter V will not vote.

Now suppose that two parties are vying for power. Party A must introduce its candidates and their positions on this 1 to 100 spectrum. Party B then introduces its candidates and their positions.

The elections work in two rounds. At the end of the first round, the two candidates who receive the most votes face each other in a runoff. If several candidates tie, then the election commission arranges the second round between members of different parties when possible. For example, if party X has one candidate that wins the most votes in the first round and several other candidates tie for receiving the second highest number of votes, then the election commission will choose a candidate from a party other than party X.

Warm-Up: Can both parties guarantee to having at least one candidate in the run-off if each puts forth two candidates? Please remember that if there is a tie between candidates from the different parties, the commission will ensure two parties are represented in the final election.



Solution to Warm-Up: Let's say that party A puts its candidates at position 25 and 75. If party B puts a candidate at 24 and 26, then B will capture voters having positions 1–24 and 26–49. Party A will capture 51–100. So, the runoff will be between the party A candidate at 75 and one of the party B candidates chosen at random by the commission. Facing party A's choice of 25 and 75, party B cannot prevent party A from having at least one candidate reach the runoff.

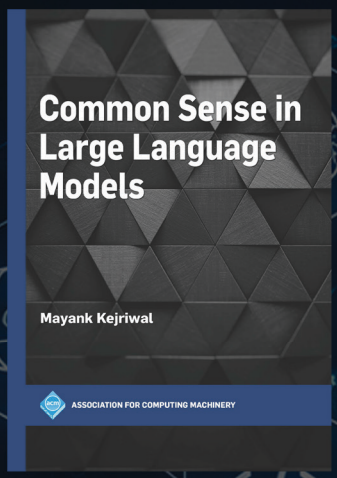
Question: Suppose there are three candidates from party B but still two from party A. Can B guarantee to have all runoff candidates be from B's party when A goes first? Prove your answer.

Solution: In that case, B can guarantee that the runoff will be between B's candidates. Let's say party A puts its candidates at X and $X+d$, respectively. One possibility is for party B to put one candidate in the middle of the A candidates (at $X + d/2$). That candidate would receive approximately $d/2$ votes. B's other candidates could go to $X-1$ and to $1+(X+d)$. This limits A's candidates to approximately a fraction $d/4$ of the votes. Provided $d/4 < X-1$ or $d/4 < 100-(X+d)$, B's candidate in the middle and one of B's other candidates will enter the runoff. We will call this middle-and-two-ends strategy Configuration 1.

If d is large, party B puts two candidates at $X+1$ and $X+d-1$ and the third candidate at $X+d+1$ or $X-1$ whichever gains that B candidate more votes (or at least no fewer). We will call this surround-one-candidate and inside the other candidate [CONTINUED ON P. 111]

IMAGE BY ANDRIJ BORYS ASSOCIATES. USING SHUTTERSTOCK.COM

COLLECTION 4 NOW PUBLISHING

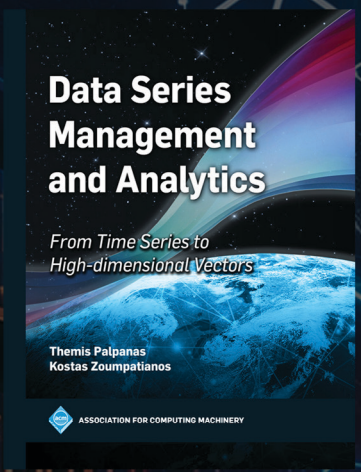


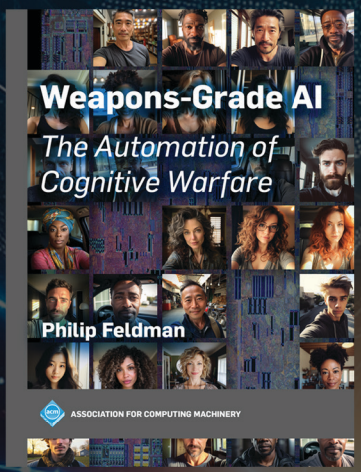
First Published!

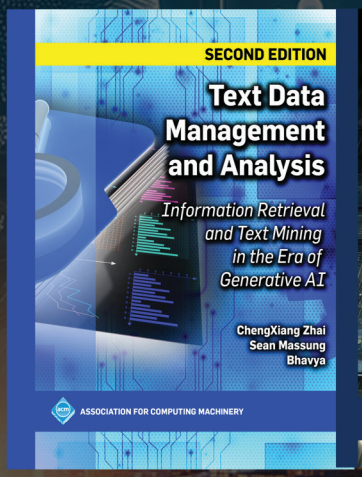
Common Sense in Large Language Models

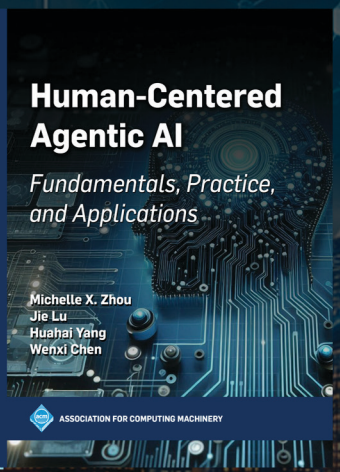
Mayank Kejriwal · ISBN: 979-8-4007-3198-3 · DOI: 10.1145/3789270

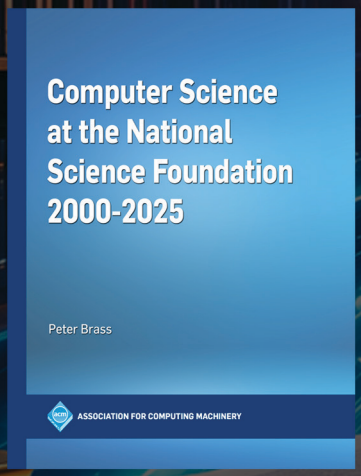
This book explores what "common sense" truly means in cognitive science and AI, why today's most advanced systems succeed in some contexts yet break down in others, and how human-like reasoning is structured—from naïve physics and causality to social and situational inference. It also examines how common sense can be systematically evaluated in non-human intelligences, highlighting emerging scientific frameworks and benchmark tests that are shaping the next generation of AI assessment.

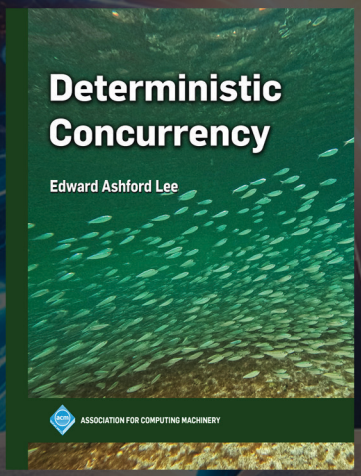












Forthcoming Titles

FROM COMPUTING FOUNDATIONS TO THE FUTURE

Collections 1, 2, and 3 are published!
Visit: https://books.acm.org

Today's Research Driving Tomorrow's Technology

The ACM Digital Library (DL) is the most comprehensive research platform available for computing and information technology and includes the ongoing contributions of the field's most renowned researchers and practitioners.

Each year, roughly 20,000 newly published articles from ACM journals, magazines, technical newsletters and annual conference volumes are added to the DL's complete full text contents of more than 550,000 articles.

The DL also features the fully integrated and comprehensive bibliographic index, *The Guide to Computing Literature*—a continually updated index featuring millions of publication records from over 5,000 publishers worldwide.

For more information, please visit
https://libraries.acm.org/
or contact ACM at
dl-info@hq.acm.org

